WEB APPLICATION DEVELOPMENT

Lab 1 Assignment Report

Project: Epic Ink Gamers Website

Student: Nguyễn Hoàng Minh Duy **ID:** ITITWE23044

Repository: https://github.com/Danxelion/Assignment1

Live URL: https://danxelion.github.io/Assignment1/

Submission Date: October 24, 2025

1. PROJECT OVERVIEW

Project Name: Epic Ink Gamers Club Website

This is a multi-page website for a gaming club focused on video game narratives and lore. The website includes 7 pages showcasing club information, activities, member profiles, a gallery, policies, and a comprehensive registration form.

2. OUTPUT & WEBSITE PAGES

2.1 Home Page (index.html)

Features:

- Title: "Epic Ink Gamers"
- Clean, minimalist design
- Video Background

Explaination:

```
<video autoplay muted loop class="bg-video">
    <source src="assets/img/haha.mp4" type="video/mp4">
</video>
```

```
.bg-video{
    position: absolute;
    width: 100%;
    height: 100%;
    overflow: hidden; /* Prevents overflow of video */
    object-fit: cover; /* Ensures video covers the area */
    z-index: 1; /* Background layer */
    opacity: 0.8; /* Slight transparency*/
}
```

The background video is configured to autoplay, loop continuously, and play muted. It employs the ‘bg-video’ CSS class for styling control. The positioning is set to `absolute`, with `width: 100%` and `height: 100%`, ensuring the video element fully covers the bounds of its parent container, which is typical for full-background video implementation. The `opacity` is set to 0.8 to achieve semi-transparency, combined with a dark background to mitigate visual discomfort and reduce glare caused by the luminous video content.

```
<header class="main-content">
    <img src="assets/img/icon.jpg" alt = "Epic Ink Gamers" class="bg-icon">
    <nav>
        <ul>
            <li><a href="index.html" class="current-page">Home</a></li>
            <li><a href="about.html">About</a></li>
            <li><a href="activities.html">Activities</a></li>
            <li><a href="member.html">Members</a></li>
            <li><a href="contact.html">Contact</a></li>
            <li><a href="gallery.html">Gallery</a></li>
            <li><a href="policy.html">Policy</a></li>
        </ul>
    </nav>
</header>
```

```css
nav ul {
    display: flex;
    flex-direction: row;
    justify-content: center;
    list-style: none; /*no bullets*/
    padding: 0;
    margin: 0;
}

/*Add spacing for nav*/
nav ul li {
    margin: 0 3em;
}

nav ul li a{
    color: var(--color-text-secondary);
    background: transparent;
    text-decoration: none;
    font-size: 1em;
    text-transform: uppercase;
}

nav ul li a:hover,
nav ul li a:focus {
    color: var(--color-text-secondary);
    font-size: 1.1em;
    text-decoration: underline;
    text-decoration-thickness: 3px;
    text-underline-offset: 6px;
}
```

```css
.bg-icon {
    position: absolute;
    top: 0.4em;
    left: 1em;
    width: 64px;
    height: 64px;
    z-index: 3;
    box-shadow: 0 0 12px var(--color-glow-icon-blue);
    border: solid 1px var(--color-text-primary);
}
```

The header section was implemented and uniformly applied across all pages. The header includes seven navigation links corresponding to the specified pages. I styled the `<nav> <ul>` element by applying Flexbox layout to display list items horizontally and removed default list bullets with "list-style: none;". The navigation elements were styled minimally, with hover and focus states enhanced to display a prominent underline and a slight scaling transformation for visual emphasis, while maintaining consistent text color throughout.

The icon within the header is styled using the `bg-icon` class, which sets explicit width and height properties. It is positioned in the top-left corner of the header using absolute or relative positioning as specified and is enhanced with box-shadow and border properties for visual delineation.

For each corresponding page, the "current-page" class has been applied to the navigation element to indicate the active page. The styling for this class utilizes the same decorations as those applied during focus and hover states.

```
  <li><a href="index.html" class="current-page">Home</a></li>
  <li><a href="about.html">About</a></li>
  <li><a href="activities.html">Activities</a></li>
  <li><a href="member.html">Members</a></li>
  <li><a href="contact.html">Contact</a></li>
  <li><a href="gallery.html">Gallery</a></li>
  <li><a href="policy.html">Policy</a></li>
```

```
nav ul li a.current-page{
    color: var(--color-text-primary);
    text-decoration: underline;
    text-decoration-thickness: 3px;
    text-underline-offset: 6px;
}
```

Finally, I put <footer> and <main> with title <h1>.

```
<main>
    <h1 class="main-content">--Epic Ink Gamers--</h1>
</main>

<footer class="main-content">
    <p>&copy; 2025 Epic Ink Gamers</p>
</footer>
```

```
footer {
    background: var(--color-bg-primary);
    font-family: 'Roboto', Arial, Helvetica, sans-serif;
    color: var(--color-text-footer);
    text-align: center;
    padding: 0.05em 0;
    font-size: 1em;
    box-shadow: 5px -5px 12px 0px var(--color-glow-white-semi);
}
```

```
h1{
    font-family: "Road Rage", Arial, sans-serif;
    letter-spacing: 2px;
    text-shadow: 0 0 30px var(--color-shadow-black);
    margin: auto;
    left : 29em;
    top: 12.9em;
}
```

The `<h1>` element was manually positioned at the bottom right corner, which may be considered suboptimal from a professional standpoint; however, this approach was quick. Additionally, the CSS class "main-content" was assigned to the `<header>`, `<h1>`, and `<footer>` elements to resolve a layering issue where the video background overlapped the content. To ensure proper stacking order, the "main-content" class was assigned a `z-index: 2`, while the "bg-video" element received a `z-index: 1`, thereby maintaining content visibility above the background video. This is also the reason I created new css file, as the “index.css” is kind of chaotic but I still copied many classes from it to “styles.css”.


```css
.main-content{
    position: relative;
    z-index:2;
    color: var(--color-text-content);
}
```

2.2 About Page (about.html)

Features:

- Auto text typing
- Banner of pictures flowing from right to left

Explainations:

I assigned an ID attribute to the <h2> element to enable the JavaScript file to target it for animation purposes. Additionally, I applied styling or decoration to the <h2> element.

```html
<h2 id = "typing-text"></h2>
```

```css
.sec h2{
    font-family: 'f Frelly', Arial, sans-serif;
    font-size: 50px;
    min-height: 1.2em; /* Ensure space for the typing text even when empty */
    margin: 0.5em;
    color: var(--color-text-secondary);
}
```

Initially, I instantiated all requisite variables, including 'textElement' assigned to the element with the identifier 'typing-text', the string 'textToType' representing the content to be rendered, the integer 'charIndex' for tracking the current position within the text, and the numeric 'typingSpeed' denoting the delay interval between character outputs.

```
//Typing animation
const textElement = document.getElementById('typing-text');
const textToType = "Welcome to Epic Ink Gamers";
let charIndex = 0;
const typingSpeed = 100;
```

Secondly, I determined that incorporating multiple colors enhances visual appeal. Therefore, I implemented a function to assign randomized RGB color values to each character. The process involves generating a random number between 0 and 1 using Math.random(), scaling this value by multiplying by 105, and subsequently adding 150. This results in a value range of 150 to 255, which ensures lighter colors suitable for a dark background. Using Math.floor() on these values allows for the construction of valid RGB color strings in the format 'rgb(r, g, b)'.

```
function getRandomLightColor(){
    const r = Math.floor(150 + Math.random() * 105);//150-255 prevent dark tones
    const g = Math.floor(150 + Math.random() * 105);
    const b = Math.floor(150 + Math.random() * 105);
    return `rgb(${r}, ${g}, ${b})`;
}
```

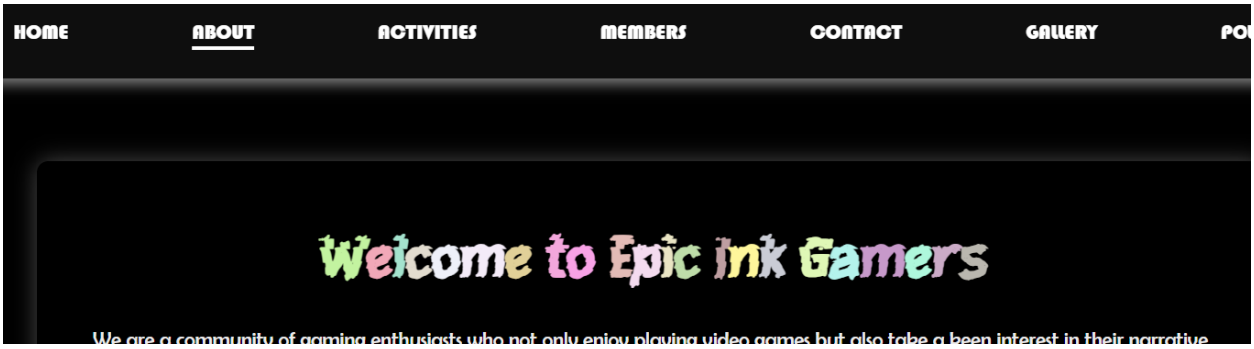
In the main function, an iterative control structure of the form `if (i < length)` is employed, with the `typeText()` at the end invoked externally to facilitate continuous character rendering. A `<span>` element is created to apply styling to the displayed text segment. The `getRandomLightColor()` function is then called to assign a dynamic color style to the span. The `appendChild()` method appends the styled span to the parent element, followed by incrementing the index `i` to proceed to the subsequent character. Finally, the typing speed is regulated using `setTimeout()`.

```
function typeText(){
    if(charIndex < textToType.length){
        const span = document.createElement('span');//Control each letter
        span.textContent = textToType[charIndex];

        span.style.color = getRandomLightColor();
        textElement.appendChild(span);//print
        charIndex++;

        setTimeout(typeText, typingSpeed);
    }
}
// Start the typing animation when the page loads
typeText();
```

The result:

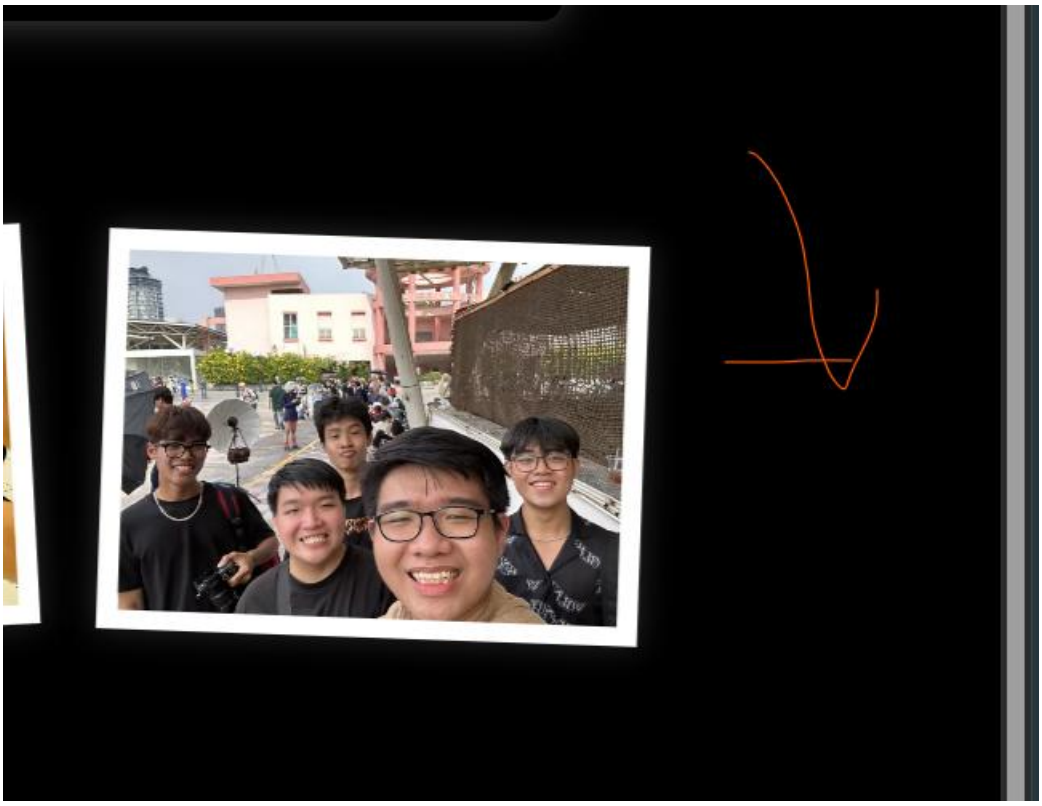


For the banner of pictures, I used css instead of js. There are 6 images, and I duplicated them once again to ensure better smooth transition when the first ones end. Each picture is managed by “polaroid-item”, and they’re controlled by “polaroid-scroll-track” staying inside “polaroid-scroll-container”.

```
<section class="polaroid-scroll-container">
    <div class="polaroid-scroll-track">
        <div class="polaroid-item"> <!--floating polaroid effect-->
            <img src="assets/img/polaroid1.png" alt="Club Event 1">
        </div>
        <div class="polaroid-item">
            <img src="assets/img/polaroid2.png" alt="Club Event 2">
        </div>
        <div class="polaroid-item">
            <img src="assets/img/polaroid3.png" alt="Club Event 3">
        </div>
        <div class="polaroid-item">
            <img src="assets/img/polaroid4.png" alt="Club Event 4">
        </div>
        <div class="polaroid-item">
            <img src="assets/img/polaroid5.jpg" alt="Club Event 5">
        </div>
        <div class="polaroid-item">
            <img src="assets/img/polaroid6.png" alt="Club Event 6">
        </div>
```

```
    </div>
 <!-- Duplicate set for seamless loop -->
    <div class="polaroid-item">
        <img src="assets/img/polaroid1.png" alt="Club Event 1">
    </div>
    <div class="polaroid-item">
        <img src="assets/img/polaroid2.png" alt="Club Event 2">
    </div>
    <div class="polaroid-item">
        <img src="assets/img/polaroid3.png" alt="Club Event 3">
    </div>
    <div class="polaroid-item">
        <img src="assets/img/polaroid4.png" alt="Club Event 4">
    </div>
    <div class="polaroid-item">
        <img src="assets/img/polaroid5.jpg" alt="Club Event 5">
    </div>
    <div class="polaroid-item">
        <img src="assets/img/polaroid6.png" alt="Club Event 6">
    </div>
 </div>
```

If I did not duplicate:



The container element is configured to establish the viewport for the banner content. The scroll-track employs a flex container with horizontal orientation to align child elements—images—in a single row, utilizing a gap property to ensure consistent spacing between items. An animation named "scroll-left" is applied, with the container’s width set to 'fit-content' to accommodate all child elements without wrapping. The hover state triggers a pause in the animation. The "@keyframes" rule defines the "scroll-left" animation, transitioning the transform property from "translateX(0)" at 0% to "translateX(-50%)" at 100%, effectuating a leftward movement of the images. Within each "polaroid-item," the "flex-shrink: 0" property prevents the images from shrinking upon container resize. Additional styling includes padding to emulate polaroid frames and a subtle box-shadow to enhance visual appeal.

```css
.polaroid-scroll-container {
    width: 100%;
    overflow: hidden;
    margin: 4em 0;
    padding: 2em 0;
    position: relative;
}

.polaroid-scroll-track {
    display: flex;
    gap: 2em;
    animation: scroll-left 30s linear infinite;
    width: fit-content;
}

.polaroid-scroll-track:hover {
    animation-play-state: paused;
}
```

```css
@keyframes scroll-left {
    0% {
        transform: translateX(0);
    }
    100% {
        transform: translateX(-50%);
    }
}

.polaroid-item {
    flex-shrink: 0;
    width: 250px;
    height: 180px;
    background-color: var(--color-border-white);
    padding: 10px;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3),
                0 0 15px rgba(255, 255, 255, 0.2);
```

The CSS selector ".polaroid-item img" defines the dimensions and presentation of the images. Additionally, a rotation transformation of 12 degrees is applied to each polaroid element to ensure visual distinctiveness.

```css
.polaroid-item img{
    width: 100%;
    height: 100%;
    object-fit: cover;
    display: block; /* Removes extra space below image */
}

.polaroid-item:hover {
    transform: scale(1.1) rotate(0deg);
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.5),
                0 0 25px rgba(255, 255, 255, 0.4);
    z-index: 10;
}
```

```css
/* Add varied rotations to polaroids */
.polaroid-item:nth-child(1) { transform: rotate(-5deg); }
.polaroid-item:nth-child(2) { transform: rotate(3deg); }
.polaroid-item:nth-child(3) { transform: rotate(-2deg); }
.polaroid-item:nth-child(4) { transform: rotate(4deg); }
.polaroid-item:nth-child(5) { transform: rotate(-3deg); }
.polaroid-item:nth-child(6) { transform: rotate(2deg); }
.polaroid-item:nth-child(7) { transform: rotate(-5deg); }
.polaroid-item:nth-child(8) { transform: rotate(3deg); }
.polaroid-item:nth-child(9) { transform: rotate(-2deg); }
.polaroid-item:nth-child(10) { transform: rotate(4deg); }
.polaroid-item:nth-child(11) { transform: rotate(-3deg); }
.polaroid-item:nth-child(12) { transform: rotate(2deg); }
```

The result:

2.3 Activities Page (activities.html)

Features:

- Embedded YouTube videos showing club gaming sessions
- Image gallery of favorite well-written game characters with captions

Explaination:

The YouTube videos were embedded by assigning their URLs to the `src` attribute of `<iframe>` elements. A CSS class named “iframe-container” was defined to control and adjust the positioning and layout of these embedded iframes.

```
<section>
    <h1>Our funny videos:</h1>
        <div class="iframe-container">
            <iframe width="560" height="315" src="https://www.youtube.com/embed/UEomlGV3iB4?si=UnlGvduSdYICSd40" title="YouTube video player" framebor
        </div>

        <div class="iframe-container">
            <iframe width="560" height="315" src="https://www.youtube.com/embed/VjKYhYiNU_k?si=WTIehMJcukQlZTh0" title="YouTube video player" framebor
        </div>
</section>
```

```
.iframe-container {
    max-width: 700px;
    margin: 0 auto 4em;
}

.iframe-container iframe {
    width: 100%;
    height: 400px;
    display: block;/* Ensures margin:auto works properly */
    border: none;
}
```

The image gallery was implemented using a <figure> element nested within a <table> element. The <figure> element ensures consistent image positioning and enforces a fixed dimension across all images. A CSS rule applying `border-radius: 8px` was employed to achieve rounded corners, complemented by a `box-shadow` property for decorative effect. The <figure> within the <table> serves to optimize positional alignment between individual images and the table structure. The <figcaption> element governs the vertical spacing between each caption and its corresponding image. Additionally, a class applied to the <figcaption> within the table

manages the spatial relationship between neighboring captions. The <table> element is styled with `margin: auto` to center within the container.

```css
figure {
    margin: 1.5em auto;
    text-align: center;
}

figure img {
    width: 400px;
    height: 500px;
    object-fit: cover;/* Crop edges instead of stretching */
    object-position: center;
    border-radius: 8px;
    box-shadow: 0 0 12px var(--color-border-white);
}


figcaption {
    font-size: 1em;
    color: var(--color-text-primary);
    margin-top: 0.5em;
    text-align: center;
```

```css
table {
    margin: auto;
}


table figure {
    border: none;
    margin: 0 3em 1em;
    text-align: center;
}


table figcaption{
    border: none;
    margin: 0 7em 3em;
```

```
<section>
    <h1>Our favorite well-written characters:</h1><br>
    <table>
        <tr>
            <td>
                <figure>
                    <img src="assets/img/james.jpg">
                </figure>
                <figcaption>James Sunderland in <i>Silent Hill 2</i></figcaption>
            </td>
            <td>
                <figure>
                    <img src="assets/img/joshua.jpg">
                </figure>
                <figcaption>Joshua Graham in <i>Fallout: New Vegas</i></figcaption>
            </td>
        </tr>
        <tr>
            <td>
                <figure>
                    <img src="assets/img/charlotte.jpg">
                </figure>
                <figcaption>Charlotte Wiltshire in <i>Hello Charlotte</i></figcaption>
            </td>
            <td>
                <figure>
                    <img src="assets/img/morgana.jpg">
```

2.4 Members Page (member.html)

Displays a roster of club members with associated relevant information. The presentation resembles that of activities.html; however, the current member list in club contains only the administrator (me). To simulate member entries, a placeholder image generated via Gemini has been utilized. Furthermore, hyperlinks to the personal websites of other students (not a part of this product – just more like an advertisement) in the same course, which are my friends, have been embedded within the list. (So, I guess you can even rate their product without opening their submissions.).

```
<section class="sec">
    <h2>Affiliate</h2>

    <table>
        <tr>
            <td>
                <figure>
                    <img src="assets/img/dang.jpg">
                </figure>
                <figcaption><a href = "https://heomaptv123.github.io/">GiGaRu</a><br>
                Phạm Vũ Hải Đăng</figcaption>
            </td>
            <td>
                <figure>
                    <img src="assets/img/long.jpeg">
                </figure>
                <figcaption><a href="https://nxtsukii.github.io/">Natsuki</a><br>
                Huỳnh Ngọc Bảo Long</figcaption>
            </td>
        </tr>
        <tr>
            <td>
                <figure>
                    <img src="assets/img/lam.jpg">
                </figure>
                <figcaption><a href = "https://liamdgg.github.io/Personal-Page/index.html">Aaron the Duck</
```

2.5 Contact Page (contact.html)

Features:

- Full form

Explaination:

I implemented all the form types specified in the requirements. Each input element was appropriately labeled, and semantic grouping was achieved using `<fieldset>` and `<legend>` tags. For personal data fields, the `required` attribute was applied to enforce validation. The form's `action` attribute is set to `#` to prevent default submission behavior, given the absence of backend processing and database integration.

```
<form class="center-form" action="#" method="post">
    <fieldset>
        <legend>Member Registration Form</legend>

        <label for="fullname">Full Name:</label>
        <input type="text" id="fullname" name="fullname" placeholder="Your full name" required>

        <label for="email">Email Address:</label>
        <input type="email" id="email" name="email" placeholder="example@gmail.com" required>

        <label for="phone">Phone Number:</label>
        <input type="tel" id="phone" name="phone" pattern="[0-9]{10}" placeholder="1234567890" required>

        <label for="password">Password:</label>
        <input type="password" id="password" name="password" minlength="6" required>

        <label for="age">Age:</label>
        <input type="number" id="age" name="age" min="13" max="120" step="1" required>

        <label for="birthdate">Date of Birth:</label>
        <input type="date" id="birthdate" name="birthdate" required>

        <label for="satisfaction">Club Satisfaction (1-10):</label>
        <input type="range" id="satisfaction" name="satisfaction" min="1" max="10" step="1">
```

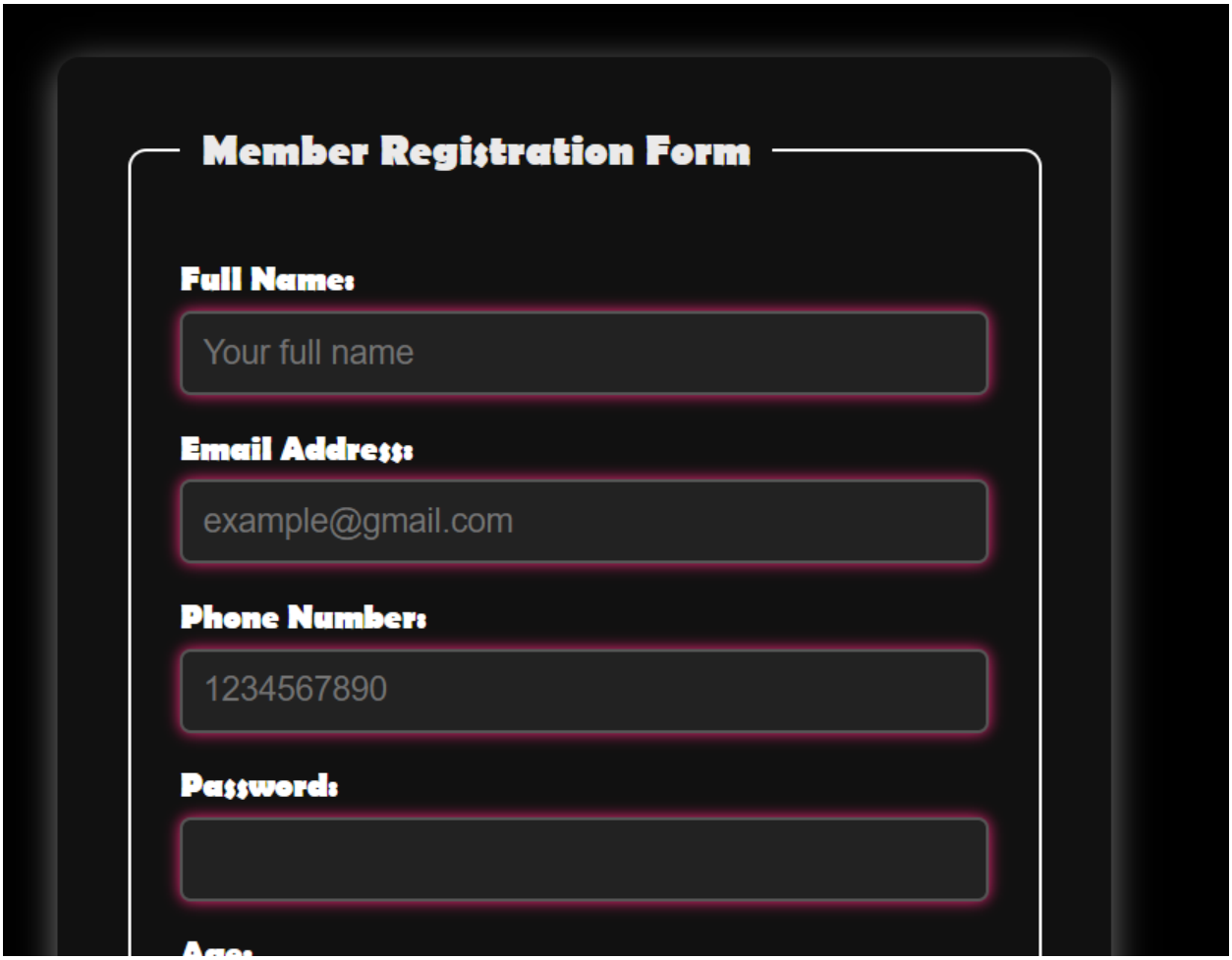
Additionally, I implemented a "center-form" layout to manage the form's structure, utilizing Flexbox with a column-oriented flex-direction for optimal alignment. The layout includes stylistic enhancements such as box-shadow effects to improve visual presentation. I defined specific CSS classes for required input fields, enabling them to display a pink glow to clearly indicate mandatory fields. Furthermore, I applied custom styles to both the <fieldset> and <legend> elements for enhanced visual distinction.

```css
.center-form {
    max-width: 500px;
    margin: 3em auto;
    padding: 2em;
    background: var(--color-form-bg);
    color: var(--color-text-secondary);
    border-radius: 12px;
    display: flex;
    flex-direction: column;
    box-shadow: 0 4px 20px var(--color-glow-white-semi);
}

input:required,
textarea:required,
select:required {
    border: 2px solid var(--color-accent-pink);
    background-color: var(--color-form-bg);
    box-shadow: 0 0 8px var(--color-accent-pink);
}
```

```css
fieldset {
    border: 2px solid var(--color-border-white);
    border-radius: 10px;
    padding: 1.5em;
}

legend {
    color: var(--color-text-primary);
    font-weight: bold;
    font-size: 1.3em;
    padding: 0 0.5em;
}
```



I modified the input components to include focus styling that perceives a blue glow upon activation. Additionally, I implemented a container element labeled "button-group" utilizing Flexbox layout to ensure the proper alignment and distribution of two button elements.

```css
.center-form input,
.center-form textarea,
.center-form select {
    width: 100%;
    padding: 0.6em;
    margin-top: 0.3em;
    border: 2px solid var(--color-border-subtle);
    border-radius: 6px;
    background: var(--color-input-bg);
    color: var(--color-text-secondary);
    font-size: 1em;
    box-sizing: border-box;
}

input[type="radio"],
input[type="checkbox"] {
    width: auto;
    margin-right: 0.5em;
}

.center-form input:focus,
.center-form select:focus,
.center-form textarea:focus {
    box-shadow: 0 0 8px var(--color-accent-blue);
    outline: none;
}
```

```css
.button-group {
    margin-top: 1.5em;
    display: flex;
    justify-content: space-between;
}

button {
    background-color: var(--color-bg-primary);
    color: var(--color-text-secondary);
    font-weight: bold;
    border: none;
    padding: 0.8em 2em;
    border-radius: 8px;
    cursor: pointer;
    transition: transform 0.2s, background 0.2s;
}

button:hover {
    background-color: var(--color-accent-hover);
    transform: scale(1.05);
}
```

2.6 Gallery Page (gallery.html)

A collection of images organized within a standard webpage layout, comprising <header>, <main>, and <footer> elements. The placement and presentation of the images are implemented using a table structure, following the same methodology employed in activities.html.

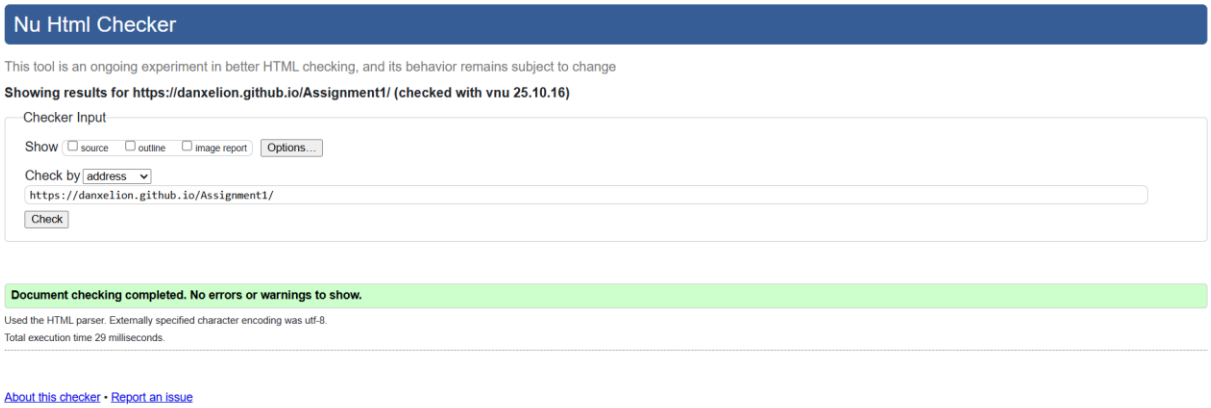
2.7 Policy Page (policy.html)

Contains official club policies and guidelines, supers imple.

3. TESTING

Testing Results:

- Website successfully deployed on GitHub Pages
- All 7 HTML pages are accessible
- CSS file loading correctly (styling applied to all pages)
- JavaScript loading properly
- Navigation links functional across all pages
- Images displaying correctly
- YouTube videos embedded and playable



4. AI USAGE LOG

- **Perplexity:** Performing investigation and data collection pertaining to information architecture, bug identification, and validation of reference web structures consistent with my conceptual design. Assisting in the organization of CSS codebases, supporting the development of systematically formatted documentation reports, and suggesting conceptual frameworks for website content from my idea. Providing guidance and good explaination on HTML and CSS syntax quickly, including the properties and best practices for markup and styling with sources from StackOverflow, Reddit, etc.
- **Gemini:** Providing assistance in JavaScript development, including explaining program logic and workflow, generating visual and video content, and assisting in optimizing code performance.
- Goblin Tools: Refining grammar, vocabulary, tone, and voice to improve the clarity and professionalism of both the website content and this report.
- **Jules:** Assistance with reviewing the checklist and requirements, assisting making the README.txt file, and managing GitHub repository tasks.