**Front End Engineering II**

**Project Report**

**Semester-IV (Batch-2022)**

**Blurry Loading using Tailwind CSS**

A red and white sign

Description automatically generated with low confidence

| **Supervised By:**  **Mr.Raveesh Samkaria** | **Submitted By:**  **Brahmjot Singh Tatla**  **Roll Number: 2210931003**  **Group: 14** |
| --- | --- |

**Department of Computer Science and Engineering**

**Chitkara University Institute of Engineering & Technology,**

**Chitkara University, Punjab**

# **Abstract:**

In the realm of web development, user experience holds paramount importance. Blurry loading animations have emerged as a popular technique to engage users during page load times, providing visual feedback and enhancing perceived performance. This abstract explores the implementation of a blurry loading animation using HTML, Tailwind CSS, and JavaScript. Leveraging the simplicity and flexibility of HTML, the elegance of Tailwind CSS, and the interactivity of JavaScript, this approach aims to deliver a seamless loading experience. The combination of these technologies allows for the creation of a dynamic and visually appealing loading animation that captivates users while content loads in the background. Through a concise examination of the implementation process and key code snippets, this abstract offers insights into how developers can integrate blurry loading animations into their web projects to elevate user experience and improve overall satisfaction.

# **Introduction:**

In the fast-paced world of web development, where attention spans are short and user expectations are high, creating a seamless and engaging user experience is crucial. One area that has garnered increasing attention is the loading experience of web pages. Traditional loading spinners and progress bars, while functional, often fail to captivate users or provide meaningful feedback during the loading process.

To address this gap, developers have turned to innovative techniques such as blurry loading animations. Blurry loading animations not only indicate that content is being loaded but also provide a visually appealing transition that keeps users engaged while they wait. This technique has become particularly popular due to its ability to enhance perceived performance and overall user satisfaction.

Throughout this exploration, we'll examine the rationale behind using blurry loading animations, the benefits they offer, and how they can be implemented effectively in web projects. By understanding the principles and techniques involved, developers can elevate the loading experience of their websites and contribute to a more enjoyable browsing experience for users.

# **Objective:**

The objective of this blurry loading project is to enhance the user experience during page loading by implementing a visually appealing and engaging loading animation. The project aims to achieve the following objectives:

* Improve Perceived Performance: By implementing a blurry loading animation, the project seeks to give users a sense of progress and activity, thereby improving perceived performance even during longer loading times.
* Enhance User Engagement: The project aims to captivate users' attention during the loading process by providing a visually appealing animation that maintains user interest and reduces the likelihood of abandonment.
* Provide Visual Feedback: Through the blurry loading animation, the project intends to provide clear and intuitive visual feedback that content is being loaded, helping users understand the status of the page and encouraging them to wait patiently.

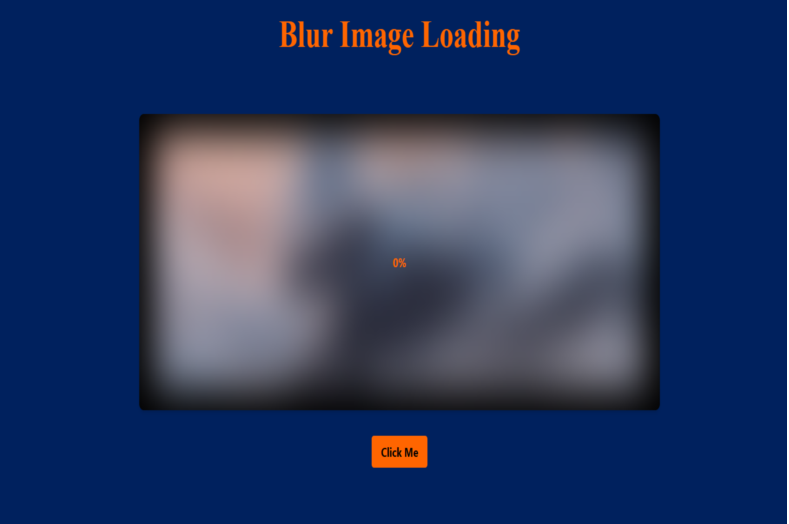
# **Significance:**

1. **Enhanced User Experience**: Implementing a blurry loading animation significantly improves the overall user experience by providing visual feedback and reducing perceived wait times. This enhanced experience can lead to higher user satisfaction and increased engagement with the website.
2. **Improved Perceived Performance**: Blurry loading animations create the illusion of continuous activity, even during longer loading times, thereby improving perceived performance. Users are more likely to perceive the website as responsive and efficient, leading to a positive impression of the brand or service.
3. **Reduced Bounce Rates**: Engaging users with a visually appealing loading animation can reduce bounce rates, as visitors are more likely to stay on the page while content loads. This can result in longer session durations and increased opportunities for conversion or interaction.
4. **Competitive Advantage**: In a competitive online landscape, providing a seamless and enjoyable user experience sets websites apart. By implementing a blurry loading animation, websites can differentiate themselves from competitors and leave a lasting impression on visitors, ultimately driving user loyalty and retention.

# **Problem Statement:**

Despite advancements in web development technologies, the loading experience of web pages remains a critical aspect that significantly impacts user engagement and satisfaction. Traditional loading indicators such as spinners or progress bars often fail to provide engaging feedback during loading times, leading to user frustration, increased bounce rates, and negative perceptions of the website.

Furthermore, as internet users increasingly expect fast and seamless browsing experiences, the need to optimize loading times and enhance perceived performance becomes paramount. However, achieving this balance between loading speed and user experience poses a challenge for developers, particularly when aiming to provide engaging visual feedback without sacrificing performance.



# 

# **Software Requirements:**

The following are the requirements for the project:

## **HTML:**

HTML serves as the foundational framework for web pages, encompassing elements enclosed within tags to delineate content and layout. Adherence to standards and validation are imperative for HTML documents, guaranteeing correct syntax, structure, and cross-browser and cross-device compatibility.

## **Tailwind :**

Tailwind CSS stands out as a prominent utility-first CSS framework, simplifying the styling of web applications. In contrast to conventional CSS frameworks dependent on pre-built components, Tailwind adopts a highly adaptable approach by furnishing an extensive library of utility classes. These classes cover a wide range of CSS properties, including margins, paddings, flexbox, and others, empowering developers to swiftly prototype and style their applications without the need for custom CSS authoring.

## **JavaScript:**

JavaScript plays a pivotal role in web development, primarily serving as a client-side scripting language. It empowers developers to craft dynamic and interactive web pages by dynamically manipulating the Document Object Model (DOM) in response to user interactions. JavaScript grants developers the capability to dynamically access and alter HTML elements and attributes, facilitating a myriad of tasks including content updates, element creation and removal, as well as styling adjustments triggered by user actions or events.

# **Project Design:**

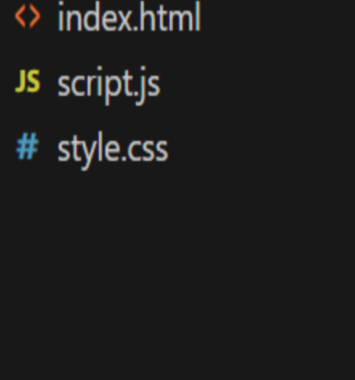
## **Project Overview:**

The objective of this project is to create a responsive website featuring a blurry loading effect using Tailwind CSS. HTML will be employed to structure the content, while Tailwind CSS will handle the visual presentation, ensuring a seamless and captivating experience for users across desktop and mobile devices.

## **File Structure:**

The project will utilize a well-organized folder structure for efficient management and future updates.

* **index.html:** Main HTML file containing the overall website structure and content.
* **style.css:** Primary CSS file for styling all website elements. It is predefined in tailwind extension.
* **script.js:** JavaScript file containing minor interactive elements.



**Methodology:**

The following shows the use of HTML, JavaScript of the project.

**HTML Code:**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta http-equiv="X-UA-Compatible" content="IE=edge">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Blur Loading Page</title>**

**<link rel="stylesheet" href="style.css">**

**<link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.0.2/dist/tailwind.min.css" rel="stylesheet">**

**<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">**

**</head>**

**<body class="font-sans text-black" style="background-color: #00215E;">**

**<!-- Responsive flex container with center alignment -->**

**<div class="flex flex-col items-center min-h-screen">**

**<h1 class="text font-bold mt-8 mb-16 text-center" style="color:#FF6500">Blur Image Loading</h1>**

**<!-- Responsive black div with flex properties -->**

**<div class="bg-black p-7 rounded-lg shadow-md relative w-full sm:max-w-xl md:max-w-2xl lg:max-w-3xl xl:max-w-4xl flex items-center justify-center overflow-hidden" style="height: 50vh;">**

**<!-- Image spans full width and is contained within black div -->**

**<img id="image" src="images/buildings-668616\_1920.jpg" alt="Image" class="rounded-lg w-full h-full object-cover" style="filter: blur(30px);">**

**<!-- Loading percentage placed to the right side of the image, centered vertically -->**

**<div id="loadPercentage" class="absolute top-1/2 transform -translate-y-1/2 font-bold" style="color: #FF6500;">**

**0%**

**</div>**

**</div>**

**<!-- The container with full screen height and centered content -->**

**<!-- Example with specific container having flex properties -->**

**<div class="flex items-center mt-8 style="height: 50vh;">**

**<button id="blurButton" class="hover:bg-red-700 text-black font-bold py-2 px-4 rounded" style="background-color: #FF6500;">Click Me</button>**

**</div>**

**</div>**

**<script src="script.js"></script>**

**</body>**

**</html>**

## **JavaScript Code:**

function blurring() {

const bg = document.getElementById('image');

const loadPercentage = document.getElementById('loadPercentage');

let load = 0;

let int = setInterval(() => {

load++;

if (load > 100) {

clearInterval(int);

loadPercentage.style.display = 'none'; // Hide the load percentage when it reaches 100%

}

bg.style.filter = `blur(${scale(load, 0, 100, 30, 0)}px`;

loadPercentage.textContent = `${load}%`; // Update the percentage dynamically

}, 30);

}

// Rest of your existing code...

const scale = (num, in\_min, in\_max,out\_min, out\_max) => {

return ((num - in\_min) \* (out\_max - out\_min)) / (in\_max - in\_min) + out\_min;

}

const button = document.getElementById("blurButton");

button.addEventListener("click", blurring);

// Function to fetch a random image from Unsplash API

function fetchRandomImage() {

fetch('https://source.unsplash.com/random/1600x900/?mountains')

.then(response => {

const imageUrl = response.url;

const bg = document.getElementById('image');

bg.src = imageUrl;

})

.catch(error => console.error('Error fetching image:', error));

}

// Call the function to fetch and display a random image from the Unsplash API

fetchRandomImage();

// Function to fetch a random image from Unsplash API when the page is reloaded

window.onload = function() {

fetchRandomImage();

};

## **CSS:**

.text {

font-size: 3rem; /\* Equivalent to 30px (1rem = 16px) \*/

font-family: 'Times New Roman', Times, serif;

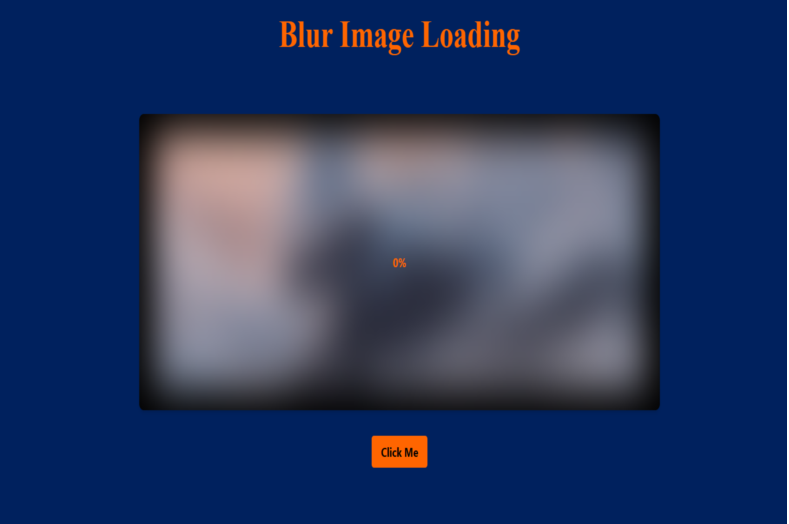
}

.h1{

font-family:'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;

}

**Results:**





# **Conclusion:**

In conclusion, the implementation of a responsive website with a blurry loading effect using Tailwind CSS represents a significant step towards enhancing user experience and engagement. By leveraging HTML for content structure and Tailwind CSS for visual presentation, the project has successfully achieved a user-friendly

# **References:**

* Tailwind CSS Documentation: <https://tailwindcss.com/docs>
* W3Schools - JavaScript: <https://www.w3schools.com/js/>
* W3Schools - HTML and CSS Tutorials: <https://www.w3schools.com/html/default.asp>