CLASSIC BOARD GAMES

PROJECT REPORT ON

**CLASSIC BOARD GAMES**

DEVELOPED BY

SWASTICA N S

BCA (DATA SCIENCE)

# INTRODUCTION

This project showcases the digital implementation of two classic board games - Tic Tac Toe and Word Scrabble - using web technologies such as HTML, CSS, and JavaScript. These games have been timeless sources of entertainment and mental exercise, enjoyed by people of all ages. By bringing them into a webbased format, the project aims to make them easily accessible and interactive across devices with modern user interfaces.

Tic Tac Toe is a simple strategy game for two players (X and O), where players take turns marking spaces in a 3×3 grid. The objective is to be the first to align three of one’s own marks horizontally, vertically, or diagonally. The game's straightforward rules make it ideal for demonstrating the basics of game logic, user interaction, and state management in JavaScript.

Word Scrabble, on the other hand, is a more complex and vocabulary-driven game where players form words from individual letters to score points. Its implementation focuses on dynamic content rendering, user input validation, and game mechanics that mimic the traditional experience.

The project was developed with a strong emphasis on responsive design, user experience (UX), and interactive gameplay. HTML provides the structure of the pages, CSS is used for styling and layout, and JavaScript powers the game logic, DOM manipulation, and real-time interaction.

This report outlines the design, functionality, and development process of both games, offering insights into the challenges faced and solutions implemented. The aim is not only to recreate these classics but also to provide a platform that is both engaging and educational from a coding perspective.

**ABOUT HTML, CSS AND JAVASCRIPT:**

The development of the classic board games—Tic Tac Toe and Word Scrabble— was accomplished using core web technologies. These include HTML, CSS, and JavaScript, which collectively enabled the creation of interactive, responsive, and user-friendly gameplay experiences directly in the browser.

## HTML (Hypertext Markup Language)

HTML is the backbone of any web application. It provides the structure of the content on a web page. In this project, HTML was used to create the layout of the game boards, buttons, input fields, and other essential elements. It ensures that all components are correctly organized and semantically appropriate.

## CSS (Cascading Style Sheets)

CSS is responsible for the presentation and design of web pages. It controls colors, layouts, fonts, and spacing. In the board games project, CSS was used to design the layout of game boards, add animations, and ensure a visually appealing and responsive design across different devices.

## JavaScript

JavaScript adds interactivity and dynamic behaviour to web pages. It enables real-time updates, user interactions, and game logic. For both Tic Tac Toe and Word Scrabble, JavaScript was the key technology used to handle game rules, turn logic, word validation, score tracking, and DOM manipulation.

# ABSTRACT

This project focuses on the development of classic board games—Tic Tac Toe and Word Scrabble—using core front-end web technologies: HTML, CSS, and JavaScript. The objective is to recreate these timeless games in a browser environment with responsive design and interactive features, providing both entertainment and a demonstration of programming concepts.

Tic Tac Toe is implemented with logic for two-player gameplay, win/draw detection, and instant feedback. Word Scrabble includes a simplified version of the traditional game with letter arrangement, word validation, and scoring mechanics. The project emphasizes clean design, intuitive gameplay, and browser compatibility without the need for external plugins or back-end systems.

This report details the technologies used, key features, and potential enhancements, showcasing how even basic web tools can be leveraged to develop engaging and functional games.

# CODE

**LOGIN PAGE - login.html**

<!DOCTYPE html>

<html>

<head>

<title>Games Zone - Login</title>

<style> body { margin: 0; padding: 0;

font-family: Arial, sans-serif;

background: linear-gradient(120deg, #2980b9, #8e44ad); height: 100vh; display: flex; flex-direction: column;

}

.header { display: flex; justify-content: flex-end; padding: 20px;

background: rgba(255, 255, 255, 0.1);

}

.header button { margin-left: 15px; padding: 8px 15px; border: none; border-radius: 5px; background: #fff; cursor: pointer;

transition: 0.3s;

}

.header button:hover {

background: #e0e0e0;

}

.container {

display: flex;

flex-direction: column; align-items: center; justify-content: center; flex-grow: 1;

}

.login-box {

background: rgba(255, 255, 255, 0.9); padding: 40px; border-radius: 10px; box-shadow: 0 0 20px rgba(0, 0, 0, 0.2);

text-align: center;

width: 300px;

} h1 { color: #333;

margin-bottom: 30px;

}

.login-button { padding: 10px 20px; background: #2980b9; color: white; border: none; border-radius: 5px; cursor: pointer; font-size: 16px;

transition: 0.3s;

}

.login-button:hover {

background: #3498db;

} .modal { display: none; position: fixed; top: 0;

left: 0; width: 100%; height: 100%; background: rgba(0, 0, 0, 0.5);

}

.modal-content { background: white; padding: 30px; border-radius: 10px; position: absolute; top: 50%; left: 50%;

transform: translate(-50%, -50%);

width: 300px;

}

.modal input { width: 100%; padding: 10px; margin: 10px 0; border: 1px solid #ddd;

border-radius: 5px;

}

.modal button { padding: 10px 20px; background: #2980b9;

color: white; border: none; border-radius: 5px; cursor: pointer;

margin-top: 10px;

}

.close {

position: absolute; right: 15px; top: 10px; font-size: 20px;

cursor: pointer;

}

</style>

</head>

<body>

<div class="header">

<button onclick="showRules()">Rules</button>

<button onclick="showAbout()">About</button>

</div>

<div class="container">

<div class="login-box">

<h1>Welcome to Games Zone</h1>

<button class="login-button" onclick="showLoginModal()">Login</button>

</div>

</div>

<div id="loginModal" class="modal">

<div class="modal-content">

<span class="close" onclick="closeModal()">&times;</span>

<h2>Enter Email</h2>

<input type="email" id="email" placeholder="Enter your email">

<a href="file:///C:/Users/Swastika/Desktop/index.html" onclick="submitEmail()">Submit</a>

</div>

</div>

<script>

function showLoginModal() {

document.getElementById('loginModal').style.display = 'block';

}

function closeModal() { document.getElementById('loginModal').style.display = 'none';

}

function submitEmail() {

const email = document.getElementById('email').value; if (email) {

alert('Email submitted successfully!');

closeModal();

} else {

alert('Please enter a valid email');

}

}

function showRules() { alert('Game Rules:\n\n1. Be respectful to other players\n2. No cheating allowed\n3. Have fun!');

}

function showAbout() {

alert('About Games Zone:\n\nWelcome to Games Zone, your ultimate destination for online gaming entertainment! We offer a wide variety of games for all ages. Have fun!!!'); }

window.onclick = function(event) {

const modal = document.getElementById('loginModal'); if (event.target == modal) { closeModal();

}

}

</script>

</body>

</html>

## INDEX PAGE – index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Classic Board Games</title>

<link rel="stylesheet" href="stylesheet.css"> </head>

<body>

<div class="container">

<header>

<h1>Classic Board Games</h1>

<p class="subtitle">Challenge yourself with these timeless favorites</p> </header>

<!-- Tic Tac Toe Card -->

<div class="game-card" onclick="showGameDetails('tictactoe')">

<span class="difficulty">Difficulty: Easy</span>

<div class="game-image">

<img src="C:\Users\Swastika\Downloads\th (1).jpg" alt="Tic Tac Toe Game">

</div>

<div class="game-info">

<h2 class="game-title">Tic Tac Toe</h2>

<p class="game-description">Classic game of X's and O's. Get three in a row horizontally, vertically, or diagonally to win!</p>

<a href="file:///C:/Users/Swastika/Desktop/project/project.html" class="playbutton">Play Now</a>

</div>

</div>

<!-- Word Scramble -->

<div class="game-card" onclick="showGameDetails('wordscramble')">

<span class="difficulty">Difficulty: Easy</span>

<div class="game-image">

<img src="C:\Users\Swastika\Downloads\th (2).jpg" alt="Word Scramble

Game">

</div>

<div class="game-info">

<h2 class="game-title">Word Scramble</h2>

<p class="game-description">Place the scramble letters in place and get to win.

Do it quick as the clock ticks!</p>

<a href="file:///C:/Users/Swastika/Desktop/project/word.html" class="playbutton">Play Now</a>

</div>

</div>

</div>

</div>

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

</body>

</html>

## TIC TAC TOE – tictactoe.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Tic Tac Toe</title>

<style> body {

font-family: 'Arial', sans-serif;

display: flex; flex-direction: column; align-items: center; justify-content: center; height: 100vh; margin: 0;

background-color: #f0f0f0;

}

h1 { color: #333;

margin-bottom: 20px;

}

.status { font-size: 20px; margin-bottom: 20px; height: 30px;

}

.board {

display: grid;

grid-template-columns: repeat(3, 100px); grid-template-rows: repeat(3, 100px);

gap: 10px;

margin-bottom: 20px;

}

.cell { width: 100px; height: 100px; background-color: white;

display: flex;

align-items: center; justify-content: center; font-size: 48px; font-weight: bold; cursor: pointer; border-radius: 8px; box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

transition: transform 0.2s, background-color 0.2s;

}

.cell:hover { background-color: #f9f9f9;

transform: scale(1.05);

}

.x {

color: #e74c3c;

}

.o {

color: #3498db;

}

.button-container { display: flex;

gap: 10px;

}

button { padding: 10px 20px;

font-size: 16px; background-color: #3498db;

color: white; border: none; border-radius: 5px; cursor: pointer;

transition: background-color 0.3s;

}

button:hover {

background-color: #2980b9;

}

#exit {

background-color: #e74c3c;

}

#exit:hover {

background-color: #c0392b;

}

.winner-line { position: absolute;

background-color: rgba(46, 204, 113, 0.5); z-index: 1;

}

.modal { display: none; position: fixed; top: 0; left: 0; width: 100%; height: 100%;

background-color: rgba(0, 0, 0, 0.5);

align-items: center; justify-content: center; z-index: 10;

}

.modal-content { background-color: white; padding: 20px; border-radius: 8px; text-align: center;

width: 300px;

}

.modal-buttons { display: flex; justify-content: center; gap: 10px;

margin-top: 20px;

} </style>

</head>

<body>

<h1>Tic Tac Toe</h1>

<div class="status" id="status">Player X's turn</div>

<div class="board" id="board">

<div class="cell" data-index="0"></div>

<div class="cell" data-index="1"></div>

<div class="cell" data-index="2"></div>

<div class="cell" data-index="3"></div>

<div class="cell" data-index="4"></div>

<div class="cell" data-index="5"></div>

<div class="cell" data-index="6"></div>

<div class="cell" data-index="7"></div>

<div class="cell" data-index="8"></div>

</div>

<div class="button-container">

<button id="reset">Reset Game</button>

<button id="exit">Exit Game</button>

</div>

<div class="modal" id="exitModal">

<div class="modal-content">

<h2>Exit Game?</h2>

<p>Are you sure you want to exit the game?</p>

<div class="modal-buttons">

<button id="confirmExit">Yes</button>

<button id="cancelExit">No</button>

</div>

</div>

</div>

<script>

document.addEventListener('DOMContentLoaded', () => { const board = document.getElementById('board'); const cells = document.querySelectorAll('.cell'); const status = document.getElementById('status'); const resetBtn = document.getElementById('reset'); const exitBtn = document.getElementById('exit'); const exitModal = document.getElementById('exitModal'); const confirmExitBtn = document.getElementById('confirmExit'); const cancelExitBtn = document.getElementById('cancelExit');

let currentPlayer = 'X';

let gameState = ['', '', '', '', '', '', '', '', ''];

let gameActive = true;

const winPatterns = [

[0, 1, 2], [3, 4, 5], [6, 7, 8], [0, 3, 6], [1, 4, 7], [2, 5, 8],

[0, 4, 8], [2, 4, 6]

];

function handleCellClick(e) {

const clickedCell = e.target;

const cellIndex = parseInt(clickedCell.getAttribute('data-index'));

if (gameState[cellIndex] !== '' || !gameActive) { return;

}

gameState[cellIndex] = currentPlayer; clickedCell.textContent = currentPlayer;

clickedCell.classList.add(currentPlayer.toLowerCase());

if (checkWin()) {

status.textContent = `Player ${currentPlayer} wins!`; gameActive = false;

return;

}

if (checkDraw()) {

status.textContent = "Game ended in a draw!"; gameActive = false;

return;

}

currentPlayer = currentPlayer === 'X' ? 'O' : 'X'; status.textContent = `Player ${currentPlayer}'s turn`;

}

function checkWin() {

for (let pattern of winPatterns) { const [a, b, c] = pattern;

if (gameState[a] && gameState[a] === gameState[b] && gameState[a] === gameState[c]) {

highlightWinPattern(pattern);

return true;

}

}

return false;

}

function checkDraw() {

return !gameState.includes('');

}

function highlightWinPattern(pattern) { pattern.forEach(index => {

cells[index].style.backgroundColor = 'rgba(46, 204, 113, 0.3)';

});

}

function resetGame() {

currentPlayer = 'X';

gameState = ['', '', '', '', '', '', '', '', '']; gameActive = true;

status.textContent = `Player ${currentPlayer}'s turn`;

cells.forEach(cell => { cell.textContent = ''; cell.classList.remove('x', 'o');

cell.style.backgroundColor = 'white';

});

}

function showExitConfirmation() {

exitModal.style.display = 'flex';

}

function hideExitConfirmation() {

exitModal.style.display = 'none';

}

function exitGame() {

alert("Thank you for playing Tic Tac Toe!"); hideExitConfirmation();

}

cells.forEach(cell => {

cell.addEventListener('click', handleCellClick);

});

resetBtn.addEventListener('click', resetGame); exitBtn.addEventListener('click', showExitConfirmation); confirmExitBtn.addEventListener('click', exitGame);

cancelExitBtn.addEventListener('click', hideExitConfirmation);

}); </script>

</body>

</html>

## WORD SCRABBLE – wordscrabble.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Word Scramble Game</title>

<style> body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; background-color: #f5f7fa;

display: flex; justify-content: center; align-items: center; min-height: 100vh; margin: 0;

padding: 20px;

}

.container { background-color: white; border-radius: 12px;

box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1); padding: 30px; width: 100%; max-width: 500px;

text-align: center;

}

h1 { color: #344563; margin-top: 0;

margin-bottom: 30px;

}

.score-container {

display: flex;

justify-content: space-between; margin-bottom: 25px;

font-size: 18px;

color: #5e6c84;

}

.scrambled-word {

font-size: 32px; font-weight: bold; letter-spacing: 5px; margin-bottom: 30px; color: #172b4d;

min-height: 48px;

}

.hint { font-size: 16px; font-style: italic; color: #6b778c; margin-bottom: 25px;

min-height: 24px;

}

input { width: 100%; padding: 12px; font-size: 18px; border: 2px solid #dfe1e6; border-radius: 6px; margin-bottom: 20px; box-sizing: border-box;

text-align: center;

}

input:focus { border-color: #4c9aff;

outline: none;

}

button { background-color: #0052cc; color: white; border: none; padding: 12px 24px; font-size: 16px; font-weight: bold; border-radius: 6px;

cursor: pointer; transition: background-color 0.2s; margin: 5px;

}

button:hover {

background-color: #0747a6;

}

button:disabled { background-color: #c1c7d0; cursor: not-allowed;

}

.button-container { display: flex; justify-content: center;

gap: 10px;

}

.timer { font-size: 18px; color: #5e6c84; margin-bottom: 25px;

}

.message { font-size: 16px; font-weight: bold; min-height: 24px;

margin-bottom: 15px;

}

.correct { color: #36b37e;

}

.incorrect { color: #ff5630;

}

.game-over { font-size: 24px; color: #172b4d;

margin: 20px 0;

} </style>

</head>

<body>

<div class="container">

<h1>Word Scramble Game</h1> <div class="score-container">

<div>Score: <span id="score">0</span></div>

<div>Level: <span id="level">1</span></div> </div>

<div class="timer">Time: <span id="timer">30</span>s</div>

<div class="scrambled-word" id="scrambled-word"></div>

<div class="hint" id="hint"></div>

<input type="text" id="user-input" placeholder="Type your answer here" autofocus>

<div class="message" id="message"></div>

<div class="button-container">

<button id="check-button">Check Answer</button>

<button id="hint-button">Show Hint</button>

<button id="skip-button">Skip Word</button>

</div>

</div>

<script>

const wordList = [

{ word: "JAVASCRIPT", hint: "A popular programming language for the web" },

{ word: "ALGORITHM", hint: "A step-by-step procedure for solving a problem" },

{ word: "FUNCTION", hint: "A reusable block of code" },

{ word: "VARIABLE", hint: "A container for storing data values" },

{ word: "ARRAY", hint: "A collection of elements in a specific order" },

{ word: "OBJECT", hint: "A collection of properties" },

{ word: "BOOLEAN", hint: "A data type with two values: true or false" },

{ word: "STRING", hint: "A sequence of characters" },

{ word: "NUMBER", hint: "A numeric data type" },

{ word: "UNDEFINED", hint: "A value that has not been assigned" },

{ word: "NULL", hint: "A value that represents no value" },

{ word: "ITERATION", hint: "The process of repeating a block of code" }, { word: "RECURSION", hint: "When a function calls itself" }

];

let currentWord = ""; let currentScrambledWord = "";

let currentHint = ""; let score = 0; let level = 1; let timer = 30; let timerInterval;

let hintsUsed = 0; let wordsSkipped = 0; let usedWords = [];

const scrambledWordElement = document.getElementById("scrambled-word"); const hintElement = document.getElementById("hint"); const userInputElement = document.getElementById("user-input");

const messageElement = document.getElementById("message"); const scoreElement = document.getElementById("score"); const levelElement = document.getElementById("level"); const timerElement = document.getElementById("timer"); const checkButton = document.getElementById("check-button"); const hintButton = document.getElementById("hint-button"); const skipButton = document.getElementById("skip-button"); function scrambleWord(word) {

const wordArray = word.split("");

let scrambledWord;

do {

for (let i = wordArray.length - 1; i > 0; i--) { const j = Math.floor(Math.random() \* (i + 1));

[wordArray[i], wordArray[j]] = [wordArray[j], wordArray[i]];

}

scrambledWord = wordArray.join("");

} while (scrambledWord === word);

return scrambledWord;

}

function getRandomWord() {

if (usedWords.length === wordList.length) {

usedWords = [];

}

let randomIndex;

do {

randomIndex = Math.floor(Math.random() \* wordList.length);

} while (usedWords.includes(randomIndex));

usedWords.push(randomIndex);

return wordList[randomIndex];

}

function newRound() { const wordObj = getRandomWord(); currentWord = wordObj.word; currentHint = wordObj.hint;

currentScrambledWord = scrambleWord(currentWord);

scrambledWordElement.textContent = currentScrambledWord; hintElement.textContent = ""; userInputElement.value = "";

messageElement.textContent = ""; userInputElement.focus();

clearInterval(timerInterval);

timer = 30;

timerElement.textContent = timer;

timerInterval = setInterval(() => { timer--;

timerElement.textContent = timer;

if (timer <= 0) {

clearInterval(timerInterval);

messageElement.textContent = `Time's up! The correct word was: ${currentWord}`; messageElement.className = "message incorrect"; setTimeout(newRound, 3000);

}

}, 1000);

}

function checkAnswer() {

const userAnswer = userInputElement.value.trim().toUpperCase();

if (userAnswer === currentWord) { clearInterval(timerInterval);

messageElement.textContent = "Correct! Great job!"; messageElement.className = "message correct";

const timeBonus = timer; const levelBonus = level \* 5; const hintPenalty = hintsUsed ? -5 : 0;

const pointsEarned = 10 + timeBonus + levelBonus + hintPenalty;

score += pointsEarned;

scoreElement.textContent = score;

if (score >= level \* 50) {

level++;

levelElement.textContent = level;

}

hintsUsed = 0;

setTimeout(() => { newRound();

}, 1500);

} else {

messageElement.textContent = "Incorrect! Try again.";

messageElement.className = "message incorrect";

}

}

function showHint() {

hintElement.textContent = currentHint;

hintsUsed = 1;

hintButton.disabled = true;

}

function skipWord() { clearInterval(timerInterval);

messageElement.textContent = `Skipped. The word was: ${currentWord}`; messageElement.className = "message";

wordsSkipped++;

setTimeout(() => { newRound();

hintButton.disabled = false;

}, 1500);

}

checkButton.addEventListener("click", checkAnswer);

userInputElement.addEventListener("keypress", (event) => { if (event.key === "Enter") {

checkAnswer();

}

});

hintButton.addEventListener("click", showHint);

skipButton.addEventListener("click", skipWord);

newRound();

</script>

</body>

</html>

## stylesheet.css

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

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

body {

background: #f0f2f5; min-height: 100vh;

padding: 20px;

}

.container { max-width: 1200px;

margin: 0 auto;

}

header { text-align: center; padding: 20px; margin-bottom: 40px; background: white; border-radius: 15px;

box-shadow: 0 2px 10px rgba(0,0,0,0.1);

}

h1 { color: #1a237e; font-size: 2.5rem;

margin-bottom: 10px;

}

.subtitle { color: #666;

font-size: 1.2rem;

}

.games-container {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr)); gap: 30px; padding: 20px;

}

.game-card { background: white; border-radius: 15px; overflow: hidden;

box-shadow: 0 4px 15px rgba(0,0,0,0.1); transition: transform 0.3s ease, box-shadow 0.3s ease; position: relative;

}

.game-card:hover { transform: translateY(-10px);

box-shadow: 0 8px 25px rgba(0,0,0,0.2);

}

.game-image { width: 100%; height: 250px; background-color: #e0e0e0; position: relative;

overflow: hidden;

}

.game-image img { width: 100%; height: 100%; object-fit: cover;

transition: transform 0.3s ease;

}

.game-card:hover .game-image img {

transform: scale(1.1);

}

.game-info {

padding: 20px;

}

.game-title { font-size: 1.5rem; color: #1a237e;

margin-bottom: 10px;

}

.game-description {

color: #666;

font-size: 1rem; line-height: 1.5;

margin-bottom: 15px;

}

.play-button { display: inline-block; padding: 10px 20px; background: #1a237e; color: white; text-decoration: none; border-radius: 5px;

transition: background 0.3s ease;

}

.play-button:hover {

background: #283593;

}

.difficulty { position: absolute; top: 10px;

right: 10px; background: rgba(255,255,255,0.9); padding: 5px 10px; border-radius: 15px; font-size: 0.9rem;

color: #1a237e;

}

@media (max-width: 768px) { .games-container { grid-template-columns: 1fr;

padding: 10px;

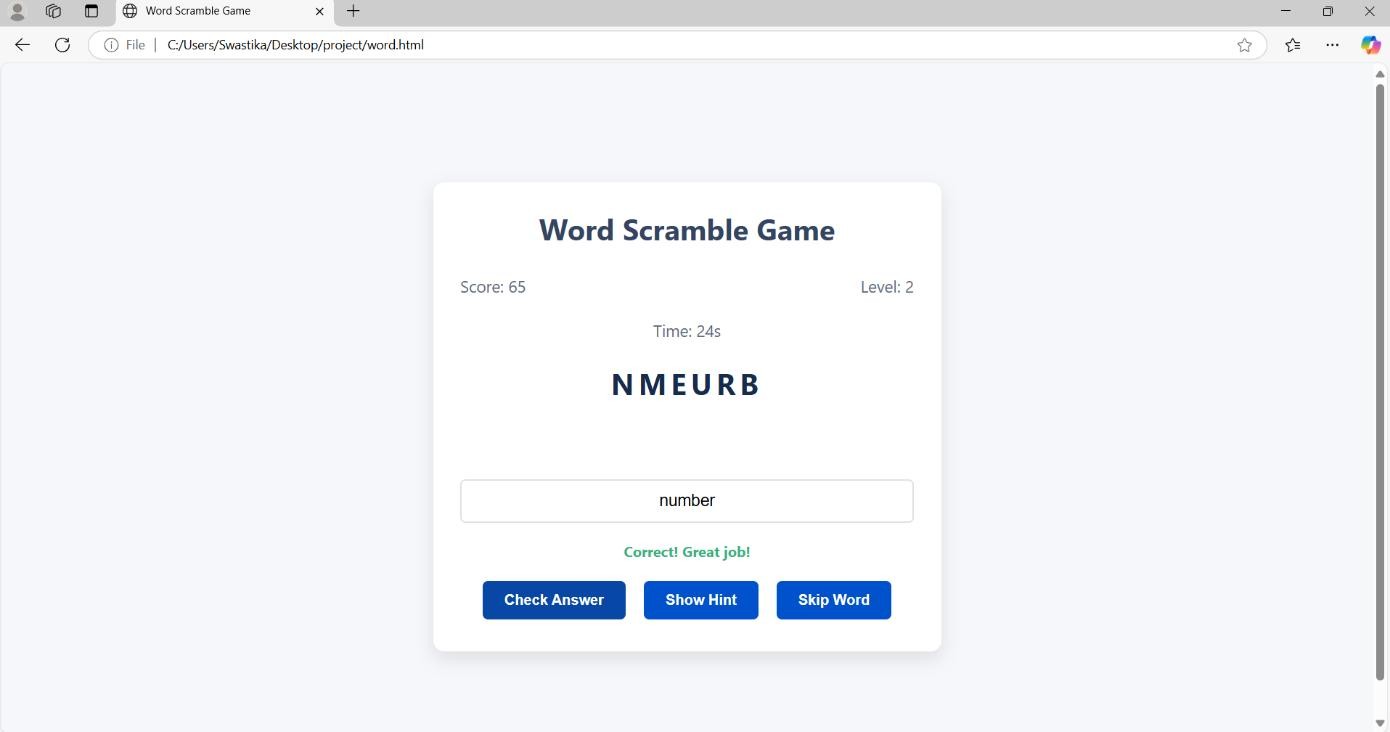
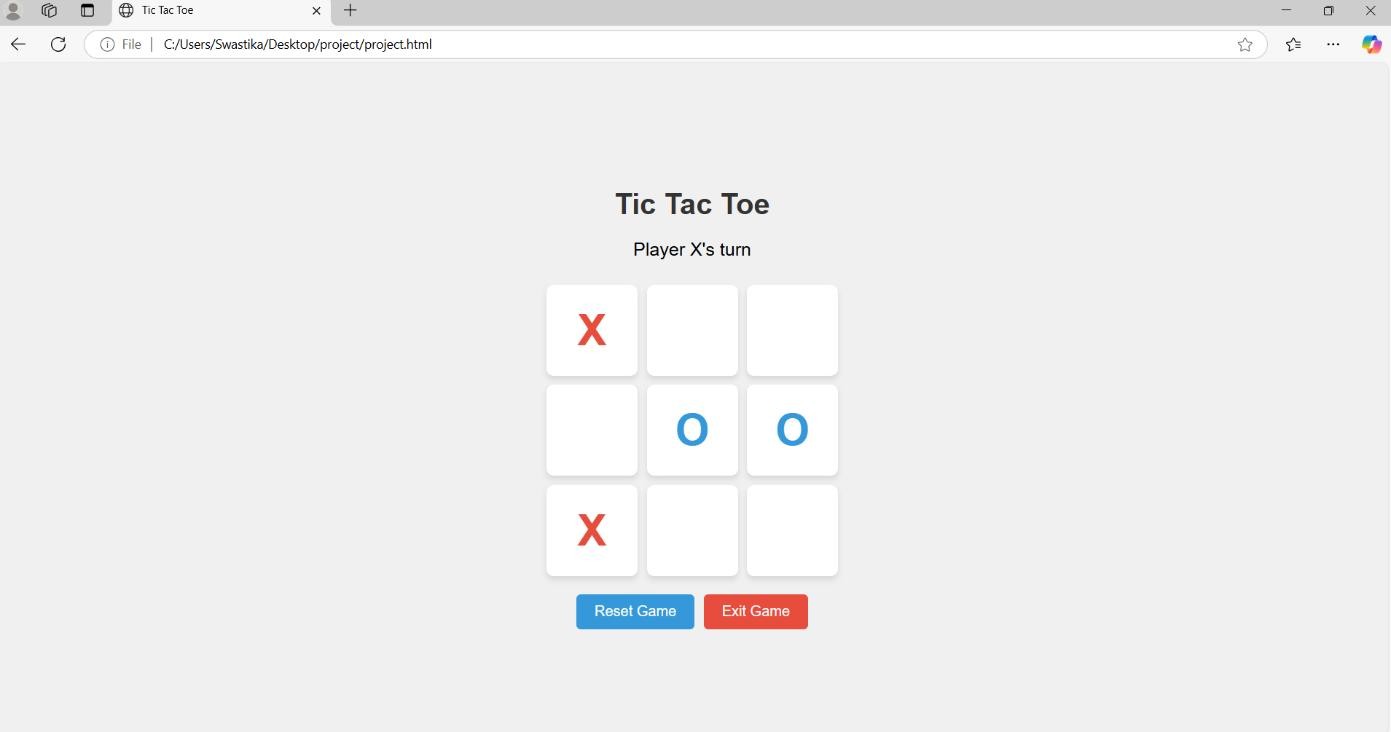
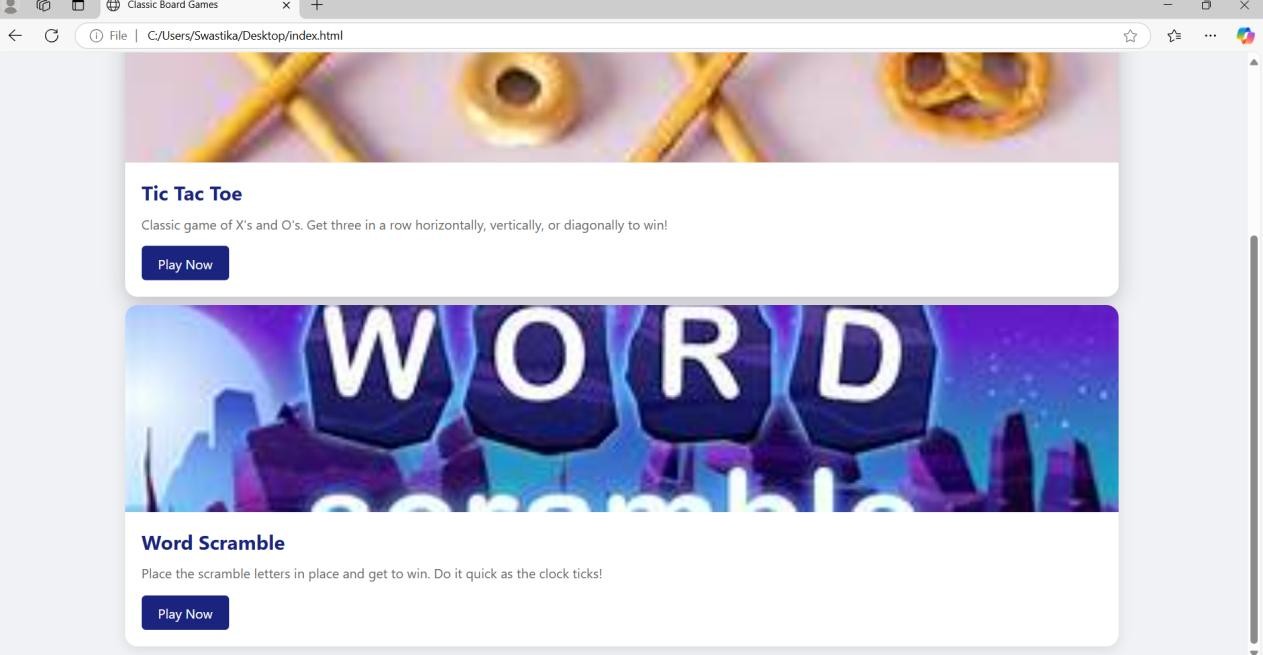
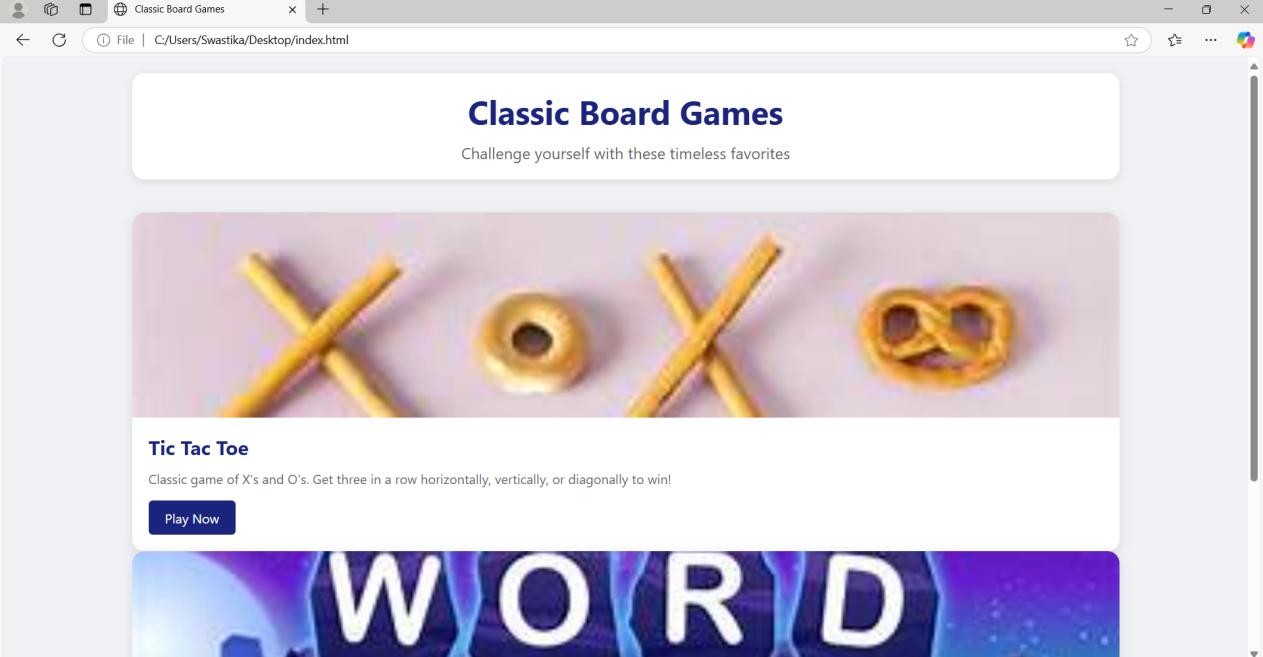
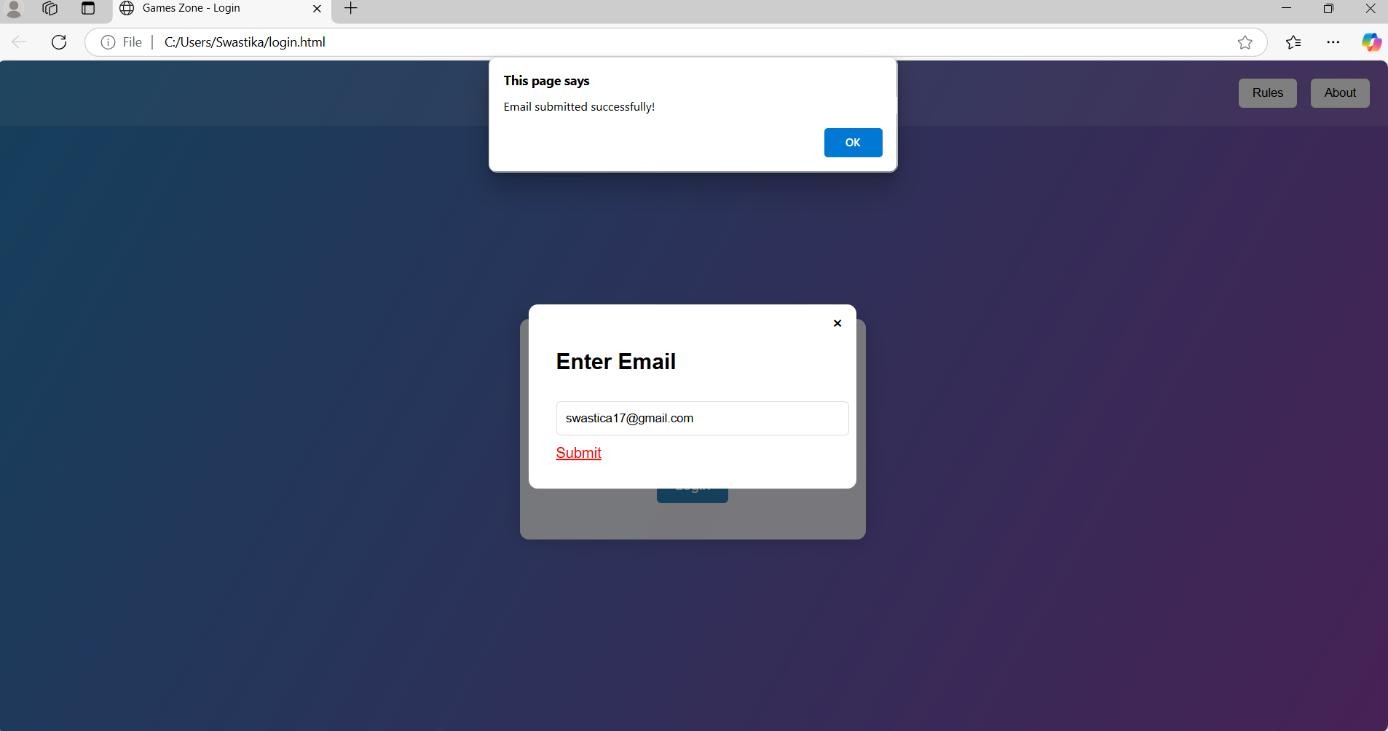
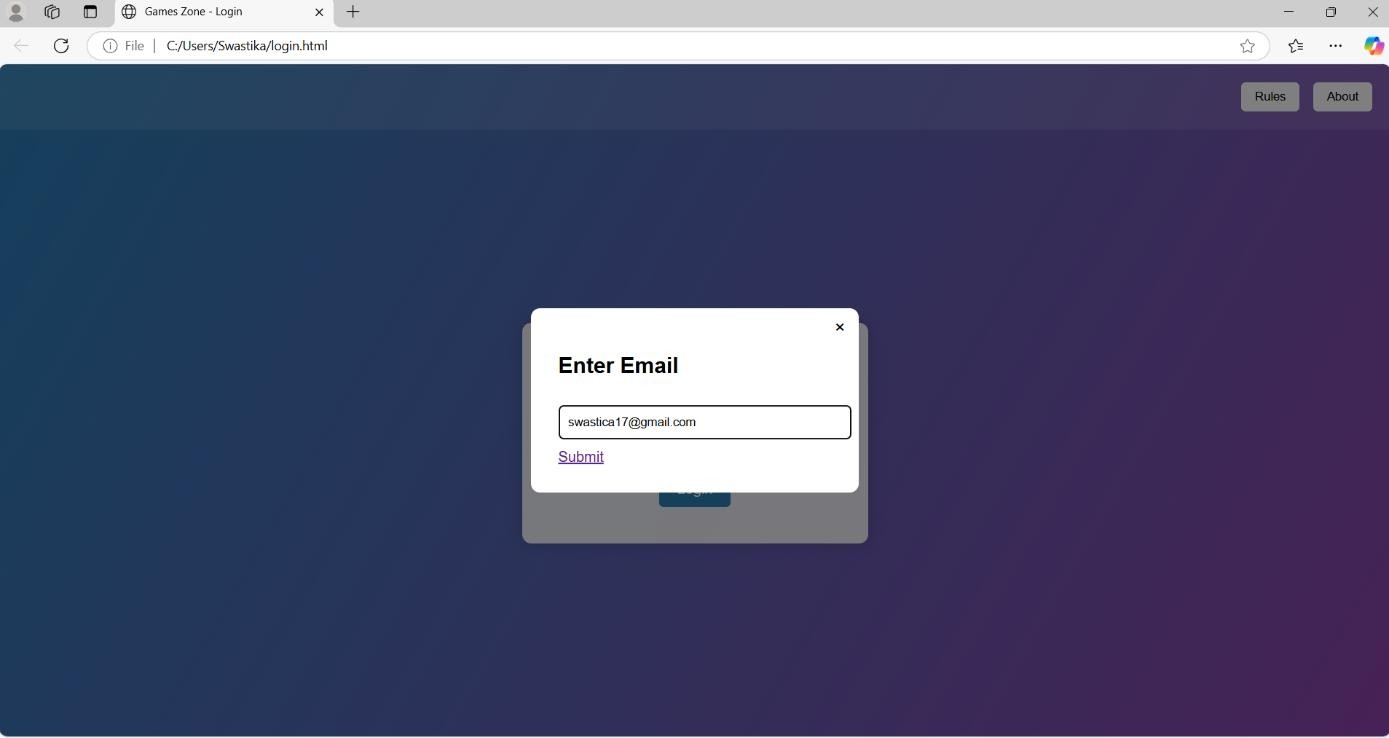
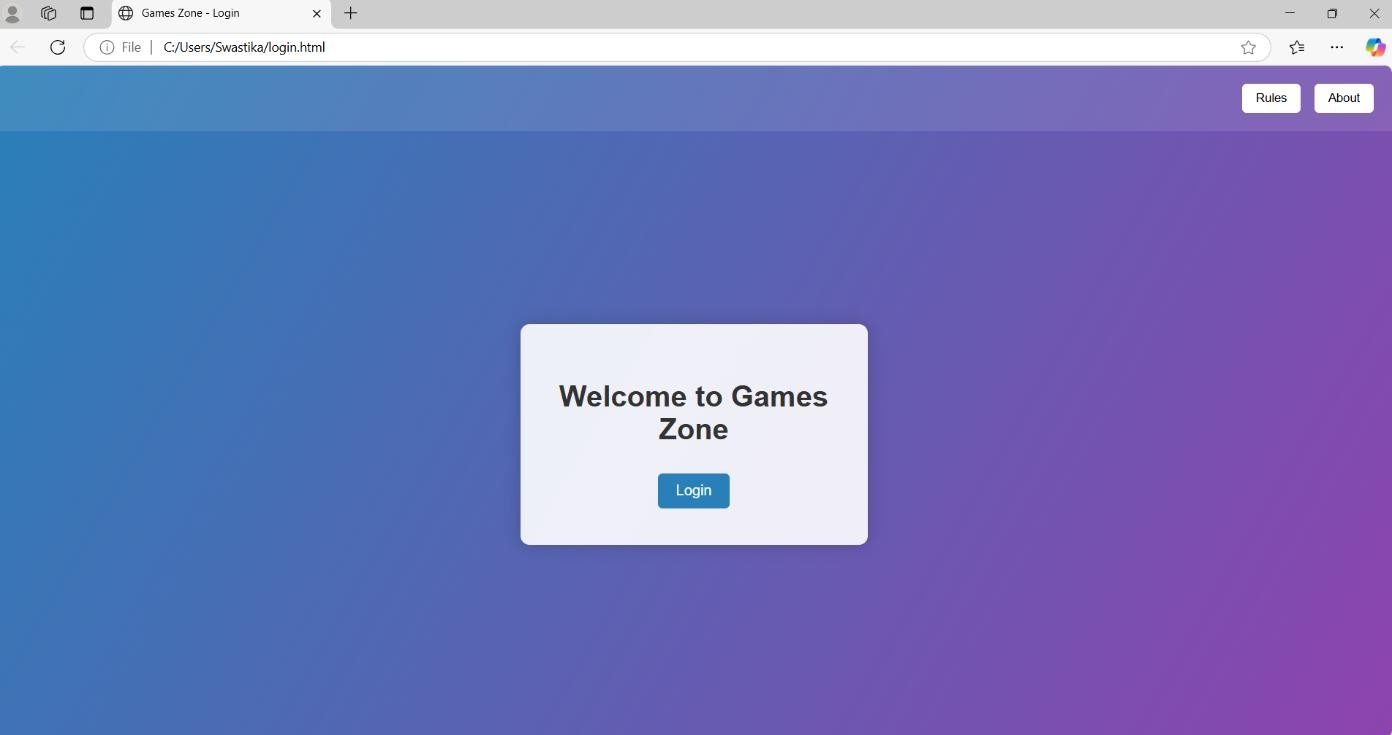
} h1 {

font-size: 2rem;

}

}

**SCREENSHOTS**



# CONCLUSION

The development of the classic board games—Tic Tac Toe and Word Scrabble— demonstrates how traditional games can be effectively brought to life using core web technologies such as HTML, CSS, and JavaScript. Through the use of structured markup, engaging design, and dynamic interactivity, the games provide a fun and intuitive user experience directly in the browser, without requiring any external software.

This project not only reinforces fundamental front-end development skills but also highlights the creative possibilities of combining logic, design, and user interaction. With future enhancements like online multiplayer, AI integration, and advanced word validation, these games can evolve into even more engaging and educational platforms.

Overall, the project successfully showcases the potential of web technologies in recreating timeless games while laying the foundation for more complex and feature-rich web applications.

# BIBLIOGRAPHY

* w3schools -<https://www.w3schools.com/html/>
* GeeksforGeeks -<https://www.geeksforgeeks.org/html-css/>
* Free code camp -<https://forum.freecodecamp.org/c/javascript/421>
* css-tricks -<https://css-tricks.com/>