**Tic-Tac-Toe Game Project Report**

**Technologies Used:** HTML, CSS, JavaScript **Difficulty Level:** Hard

**Introduction**

**1.1 Overview**

The Tic-Tac-Toe game is a two-player strategy game played on a **3x3 grid**. Players take turns marking spaces with **“X” or “O”**, aiming to form a **winning row, column, or diagonal**.

**1.2 Objectives**

* Develop an interactive, **responsive Tic-Tac-Toe game**.
* Implement **game logic** to handle moves, win conditions, and ties.
* Enhance **user experience** with dynamic UI updates.

**Page 2: Technologies Used**

**2.1 Frontend Technologies**

* **HTML**: Defines game structure.
* **CSS**: Styles the interface.
* **JavaScript**: Handles logic and interactivity.

**2.2 Key Libraries**

* **Vanilla JavaScript** (ES6 features).
* **CSS Flexbox/Grid** for responsiveness.

**Page 3: System Design**

**3.1 Game Architecture**

**Components:**

* **Game Board**: 3x3 grid.
* **Player Interaction**: Click detection on cells.
* **Win Condition Checker**: Identifies if a player wins.

**3.2 User Flow**

1. **Player clicks on a cell** → “X” or “O” appears.
2. **Game checks for a win/tie**.
3. **Display winner message** or reset for a new round.

**Page 4: HTML Structure**

**4.1 Code Example**

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>

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

</head>

<body>

<h1>Tic-Tac-Toe</h1>

<div id="game-board">

<div class="cell" id="0"></div>

<div class="cell" id="1"></div>

<div class="cell" id="2"></div>

<div class="cell" id="3"></div>

<div class="cell" id="4"></div>

<div class="cell" id="5"></div>

<div class="cell" id="6"></div>

<div class="cell" id="7"></div>

<div class="cell" id="8"></div>

</div>

<p id="status"></p>

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

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

</body>

</html>

**Page 5: CSS Styling**

**5.1 Code Example**

css

body {

text-align: center;

font-family: Arial, sans-serif;

}

#game-board {

display: grid;

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

grid-gap: 10px;

margin: 20px auto;

width: 320px;

}

.cell {

width: 100px;

height: 100px;

display: flex;

align-items: center;

justify-content: center;

font-size: 24px;

border: 2px solid black;

cursor: pointer;

}

**Page 6: JavaScript Logic**

**6.1 Core Game Logic**

js

const cells = document.querySelectorAll(".cell");

const statusText = document.getElementById("status");

const resetBtn = document.getElementById("reset");

let currentPlayer = "X";

let board = ["", "", "", "", "", "", "", "", ""];

let gameActive = true;

cells.forEach(cell => {

cell.addEventListener("click", () => handleMove(cell.id));

});

function handleMove(index) {

if (board[index] === "" && gameActive) {

board[index] = currentPlayer;

document.getElementById(index).innerText = currentPlayer;

checkWinner();

currentPlayer = currentPlayer === "X" ? "O" : "X";

}

}

function checkWinner() {

const winConditions = [

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

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

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

];

winConditions.forEach(condition => {

if (board[condition[0]] && board[condition[0]] === board[condition[1]] && board[condition[0]] === board[condition[2]]) {

statusText.innerText = `Player ${board[condition[0]]} Wins!`;

gameActive = false;

}

});

if (!board.includes("") && gameActive) {

statusText.innerText = "It's a Tie!";

gameActive = false;

}

}

resetBtn.addEventListener("click", () => {

board = ["", "", "", "", "", "", "", "", ""];

gameActive = true;

statusText.innerText = "";

cells.forEach(cell => cell.innerText = "");

});

**Page 7: Responsiveness**

**7.1 Mobile Optimization**

* **CSS Grid** automatically adjusts.
* **Media Queries** for different screen sizes.

**Page 8: User Interaction**

**8.1 Enhancements**

✅ **Clickable cells** ✅ **Win detection** ✅ **Reset button functionality**

**Page 9: Testing & Debugging**

**9.1 Test Scenarios**

* **Player clicks in a cell** → Symbol appears.
* **Game ends when win condition is met**.
* **Tie scenario correctly displayed**.

**Page 10: Conclusion & Improvements**

**10.1 Summary**

* Developed **fully functional Tic-Tac-Toe**.
* Implemented **responsive design**.
* Used **clean and structured code**.

**10.2 Future Enhancements**

✔️ **AI-based Opponent** (Random/Strategic moves). ✔️ **Score Tracking** across multiple rounds. ✔️ **Animation effects** for better UX.