**Project Report:**

**Tic Tac Toe**

**Introduction**

The Tic Tac Toe project is a classic example of a simple game often used to illustrate fundamental programming concepts.

This report provides a comprehensive overview of the project, including objectives, design, implementation, testing, and future improvements.

**Objectives**

The primary objectives of the Tic Tac Toe project are:

* To develop a functional Tic Tac Toe game that can be played between two human players or between a human player and a computer.
* To demonstrate the application of basic programming concepts such as loops, conditionals, and functions.
* To provide a user-friendly interface for an engaging gaming experience.

**Design**

**Game Rules**

* Objective:The game is played on a 3x3 grid. Players take turns to mark a cell with their respective symbols (usually 'X' and 'O'). The first player to align three of their symbols horizontally, vertically, or diagonally wins. If all cells are filled without a winner, the game ends in a draw.

**User Interface**

* Text-based: In a console-based version, the board is represented using text characters. Players input their moves as coordinates.
* Graphical: In a GUI version, the board is visually represented with clickable buttons or cells, and moves are made by interacting with the interface.

**Architecture**

* Game Board: A 3x3 grid to represent the state of the game.
* Game Logic: Handles player moves, checks for wins or draws, and alternates turns.
* User Interface: Allows users to make moves and displays the current state of the board.

**Frontend Technologies**

**HTML Structure:**

● Creating the game board with a 3x3 grid of cells for the Tic-Tac-Toe game.

● Including a display area to show the current player's turn and the game result (win,tie, or ongoing).

   <div class="wrapper">

       <div class="container">

           <button class="button-option"></button>

           <button class="button-option"></button>

           <button class="button-option"></button>

           <button class="button-option"></button>

           <button class="button-option"></button>

           <button class="button-option"></button>

           <button class="button-option"></button>

           <button class="button-option"></button>

           <button class="button-option"></button>

       </div>

       <button id="restart">Restart</button>

   </div>

   <div class="popup hide">

       <p id="message">Sample Message</p>

       <button id="new-game">New Game</button>

   </div>

**CSS Styling:**

● Styling the game board, cells, and display area to make it visually appealing.

● Using CSS for responsive design to ensure the game works on various screen sizes.

.wrapper {

    position: absolute;

    transform: translate(-50%, -50%);

    top: 50%;

    left: 50%;

  }

  .container {

    width: 70vmin;

    height: 70vmin;

    display: flex;

    flex-wrap: wrap;

    gap: 2vmin;

  }

  .button-option {

    background: #ffffff;

    height: 20vmin;

    width: 20vmin;

    border: none;

    border-radius: 8px;

    font-size: 12vmin;

    color: rgb(19, 4, 12);

    box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);

  }

  #restart {

    font-size: 1.3em;

    padding: 1em;

    border-radius: 6px;

    background-color: #ad3333;

    color: #edeff7;

    border: none;

    position: relative;

    margin: 1.6em auto 0 auto;

    display: block;

  }

  .popup {

    background: linear-gradient(135deg, #0b0714, #50032d);

    height: 100%;

    width: 100%;

    position: absolute;

    display: flex;

    z-index: 2;

    align-items: center;

    justify-content: center;

    flex-direction: column;

    gap: 1em;

    font-size: 12vmin;

  }

  #new-game {

    font-size: 0.6em;

    padding: 0.5em 1em;

    background-color: #b32b2b;

    color: #ffffff;

    border-radius: 0.3em;

    border: none;

  }

  #message {

    color: #fff9f9;

    text-align: center;

    font-size: 1em;

  }

  .popup.hide {

    display: none;

  }

**JavaScript Game Logic:**

● Implementing the game logic to allow two players to take turns.

● Checking for a win condition (three in a row, column, or diagonal) or a tie.

● Displaying the game result when a player wins or when the game ends in a tie.

● Providing a button to reset the game for a new round.

let btnRef = document.querySelectorAll(".button-option");

let popupRef = document.querySelector(".popup");

let newgameBtn = document.getElementById("new-game");

let restartBtn = document.getElementById("restart");

let msgRef = document.getElementById("message");

let winningPattern = [

  [0, 1, 2],

  [0, 3, 6],

  [2, 5, 8],

  [6, 7, 8],

  [3, 4, 5],

  [1, 4, 7],

  [0, 4, 8],

  [2, 4, 6],

];

let xTurn = true;

let count = 0;

const disableButtons = () => {

  btnRef.forEach((element) => (element.disabled = true));

  popupRef.classList.remove("hide");

};

const enableButtons = () => {

  btnRef.forEach((element) => {

    element.innerText = "";

    element.disabled = false;

  });

  popupRef.classList.add("hide");

};

const winFunction = (letter) => {

  disableButtons();

  if (letter == "X") {

    msgRef.innerHTML = "&#x1F389; <br> 'X' Wins";

  } else {

    msgRef.innerHTML = "&#x1F389; <br> 'O' Wins";

  }

};

const drawFunction = () => {

  disableButtons();

  msgRef.innerHTML = "&#x1F60E; <br> It's a Draw";

};

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

  count = 0;

  enableButtons();

});

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

  count = 0;

  enableButtons();

});

const winChecker = () => {

  for (let i of winningPattern) {

    let [element1, element2, element3] = [

      btnRef[i[0]].innerText,

      btnRef[i[1]].innerText,

      btnRef[i[2]].innerText,

    ];

    if (element1 != "" && (element2 != "") & (element3 != "")) {

      if (element1 == element2 && element2 == element3) {

        winFunction(element1);

      }

    }

  }

};

btnRef.forEach((element) => {

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

    if (xTurn) {

      xTurn = false;

      element.innerText = "X";

      element.disabled = true;

    } else {

      xTurn = true;

      element.innerText = "O";

      element.disabled = true;

    }

    count += 1;

    if (count == 9) {

      drawFunction();

    }

    winChecker();

  });

});

Conclusion

The Tic Tac Toe project successfully demonstrates fundamental programming concepts through the creation of a simple yet engaging game. The project provides a solid foundation for understanding game logic, and user interface design.

Future enhancements could expand its functionality and reach, providing a more comprehensive gaming experience.

**Thank You**