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

**Project Overview**

This project delivers a web-based Tic-Tac-Toe game designed for a single human player against a computer AI. The game aims to provide an engaging and visually appealing user experience, built with modern web technologies.

**Game Features**

* **Player vs. Computer AI:** The game allows a human player ('X') to compete against a basic computer opponent ('O').
* **Intuitive Gameplay:** Players click on empty cells to make their move.
* **Clear Status Display:** The game dynamically updates a status message indicating whose turn it is, and announces the winner or a draw.
* **Winning Condition Highlighting:** Upon a win, the three cells forming the winning line are visually highlighted.
* **Responsive Design:** The game interface is designed to adapt gracefully to various screen sizes, from mobile devices to desktop computers.
* **Restart Functionality:** A "Restart Game" button allows players to easily reset the board and begin a new game.
* **Subtle Computer Delay:** A short delay is introduced before the computer makes its move, providing a more natural and less instantaneous experience.

**Technologies Used**

* **HTML5:** Provides the basic structure and content of the web page, including the game board and controls.
* **CSS3 (with Tailwind CSS):**
  + **Tailwind CSS CDN:** Used for rapid styling and creating a modern, utility-first design. This simplifies responsive design and aesthetic enhancements like rounded corners, shadows, and spacing.
  + **Custom CSS:** Applied for gradients, specific element sizing, and hover effects to ensure an attractive and consistent visual theme.
  + **Responsive Design:** Media queries and Tailwind's responsive classes are utilized to ensure the layout adjusts optimally across different screen sizes.
* **JavaScript (ES6+):** Implements the game logic, including:
  + Managing the game board state.
  + Handling player moves and computer AI moves.
  + Determining win/draw conditions.
  + Updating the user interface dynamically.
  + Event listeners for user interactions (cell clicks, reset button).

**Game Logic (JavaScript)**

1. **board Array:** A single array of 9 elements **(['', '', ..., ''])** represents the Tic-Tac-Toe grid. Each element stores the mark ('X', 'O') or an empty string **(' ')** for an unplayed cell.
2. **currentPlayer:** A variable tracking whose turn it is, initialized to 'X' (human player).
3. **gameActive:** A boolean flag that controls whether moves can be made, set to false when the game ends (win or draw).
4. **winningConditions:** An array of arrays, defining all possible winning combinations of cell indices.
5. **initializeGame():**
   * Resets the board array to all empty strings.
   * Sets currentPlayer back to 'X' and gameActive to true.
   * Clears all marks and winning highlights from the UI cells.
   * Updates the status display to "Player X's Turn".
6. **checkGameStatus():**
   * Iterates through winningConditions to check if any player has achieved three in a row.
   * Checks if all cells are filled to determine a draw.
   * Returns an object indicating the game status ('win', 'draw', 'continue') and the winningCells indices if applicable.
7. **handleGameEnd(gameResult, lastPlayerMoved):**
   * Called when checkGameStatus() indicates a win or draw.
   * Updates the statusDisplay with the appropriate message (e.g., "Player X Wins!", "It's a Draw!").
   * Sets gameActive to false to prevent further moves.
   * Adds a win CSS class to the winning cells for visual emphasis.
8. **computerMove():**
   * Executed when it's the computer's turn (currentPlayer === 'O').
   * Identifies all empty cells on the board.
   * Randomly selects one of the available empty cells.
   * Places the computer's 'O' mark on the selected cell in the board array and updates the UI.
   * Calls checkGameStatus() and handleGameEnd() to see if the game has concluded.
   * If the game continues, it switches currentPlayer back to 'X' and updates the status display.
9. **handleCellClick(event):**
   * Triggered when a human player clicks on a cell.
   * Validates the click: ensures the cell is empty, the game is active, and it's the human's turn.
   * Places the human's 'X' mark on the clicked cell in the board array and updates the UI.
   * Calls checkGameStatus() and handleGameEnd().
   * If the game continues, it sets currentPlayer to 'O' (computer's turn), updates the status message, and then uses setTimeout(computerMove, 700) to introduce a delay before the computer's move.
10. **Event Listeners:**
    * Event listeners are attached to each game cell to trigger handleCellClick on a 'click' event.
    * An event listener is attached to the "Restart Game" button to trigger initializeGame on a 'click' event.
    * window.onload is used to call initializeGame once the page has fully loaded, ensuring the game is ready.

This project demonstrates a functional and user-friendly Tic-Tac-Toe game with a basic AI, implemented using standard web technologies.