Name: Himansu Kumar Das
Registration number: 12110286
College: lovely professional university
Email id : himansudas439@gmail.com

Tic-Tac-Toe

Objective

Our project name is Tic-Tac-Toe sport. This game could be very famous and is easy with the aid of itself. it is a two-player game. in this recreation, there isa board with n x n squares. In our recreation, it's far three x 3 squares. The purpose of Tic-Tac-Toe is to be one of the players to get 3 same symbols in a row - horizontally, vertically, or diagonally - on a 3 x three grid.

Rules of the Game

- The game is to be played between two people (in this program between HUMAN and COMPUTER).
- One of the players chooses 'O' and the other 'X' to mark their respective cells.
- The game starts with one of the players and the game ends when one of the players has one whole row/ column/ diagonal filled with his/her respective character ('O' or 'X').
- If no one wins, then the game is said to be draw.

Index.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">
    <link rel="stylesheet" href="style.css">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Itim&display=swap"
rel="stylesheet">
    <script src="./index.js"></script>
    <title>Tic-Tac-Toe</title>
</head>
<body>
    <main class="background">
        <section class="title">
            <h1>Tic Tac Toe</h1>
        </section>
        <section class="display">
            Player <span class="display-player playerX">1</span>'s turn
        </section>
        <section class="container">
            <div class="tile"></div>
```

```
            <div class="tile"></div>
            <div class="tile"></div>
            <div class="tile"></div>
            <div class="tile"></div>
            <div class="tile"></div>
            <div class="tile"></div>
            <div class="tile"></div>
            <div class="tile"></div>
        </section>
        <section class="display announcer hide"></section>
        <section class="controls">
            <button id="reset">Reset</button>
        </section>
    </main>
</body>
</html>
```

Style.css

```
* {
    padding: 0;
    margin: 0;
    font-family: 'Itim', cursive;
}

/*.background {
    background-color: black;
    height: 100vh;
    padding-top: 1px;
}
*/
.title {
    color: yellow;
    text-align: center;
    font-size: 40px;
    margin-top: 10%;
}

.display {
    color: white;
    font-size: 25px;
    text-align: center;
    margin-top: 1em;
    margin-bottom: 1em;
}

.hide {
    display: none;
```

```css
}

.container {
    margin: 0 auto;
    display: grid;
    grid-template-columns: 33% 33% 33%;
    grid-template-rows: 33% 33% 33%;
    max-width: 300px;

}

.tile {
    border: 1px solid rgb(150, 54, 54);
    min-width: 100px;
    min-height: 100px;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 50px;
    cursor: pointer;
}

.playerX {
    color: blue;
}

.playerO {
    color: black;
}

.controls {
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
    margin-top: 1em;
}

.controls button {
    color: white;
    padding: 8px;
    border-radius: 8px;
    border: none;
    font-size: 20px;
    margin-left: 1em;
    cursor: pointer;
}
```

```css
.restart {
    background-color: #498AFB;
}

#reset {
    background-color: green;
}
.line{
    color: white;
}
body{
    background-image: url("img.png");
    background-size: cover;
}
```

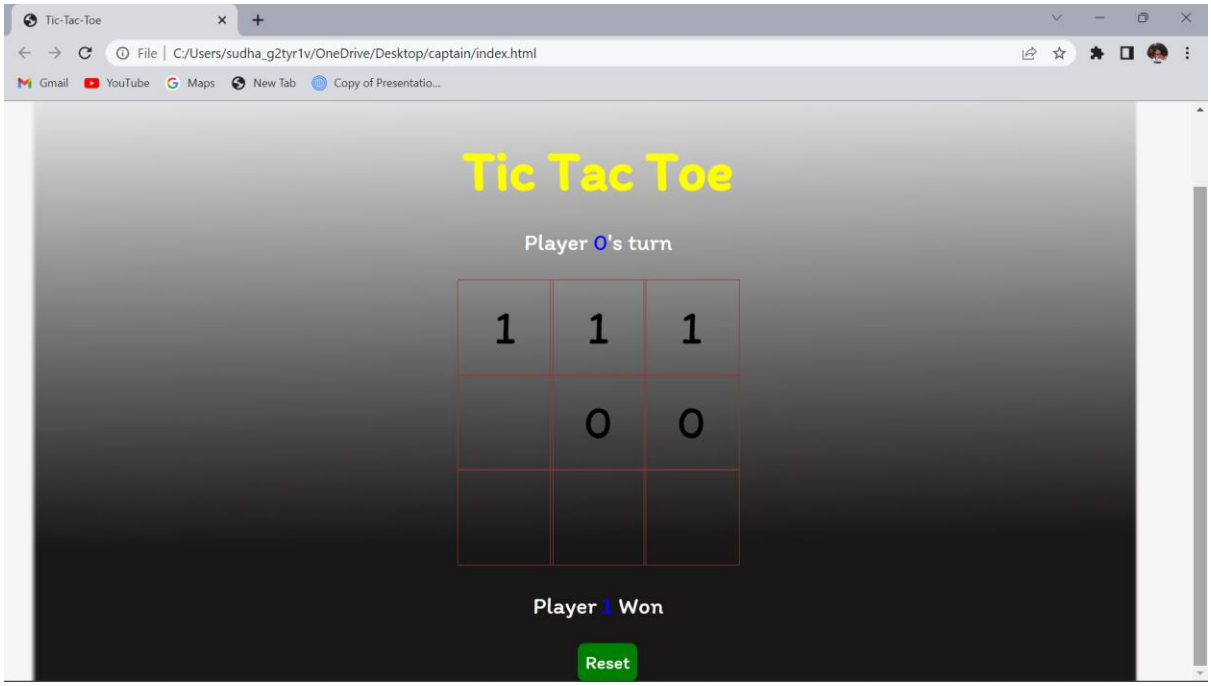
Index.js

```js
window.addEventListener('DOMContentLoaded', () => {
    const tiles = Array.from(document.querySelectorAll('.tile'));
    const playerDisplay = document.querySelector('.display-player');
    const resetButton = document.querySelector('#reset');
    const announcer = document.querySelector('.announcer');

    let board = ['', '', '', '', '', '', '', '', ''];
    let currentPlayer = '1';
    let isGameActive = true;

    const PLAYERX_WON = 'PLAYERX_WON';
    const PLAYERO_WON = 'PLAYERO_WON';
    const TIE = 'TIE';
    const winningConditions = [
        [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 handleResultValidation() {
        let roundWon = false;
        for (let i = 0; i <= 7; i++) {
            const winCondition = winningConditions[i];
            const a = board[winCondition[0]];
            const b = board[winCondition[1]];
            const c = board[winCondition[2]];
```

```
            if (a === '' || b === '' || c === '') {
                continue;
            }
            if (a === b && b === c) {
                roundWon = true;
                break;
            }
        }

    if (roundWon) {
            announce(currentPlayer === '1' ? PLAYERX_WON : PLAYERO_WON);
            isGameActive = false;
            return;
        }

    if (!board.includes(''))
        announce(TIE);
    }

    const announce = (type) => {
        switch(type){
            case PLAYERO_WON:
                announcer.innerHTML = 'Player <span class="playerO">0</span>
Won';
                break;
            case PLAYERX_WON:
                announcer.innerHTML = 'Player <span class="playerX">1</span>
Won';
                break;
            case TIE:
                announcer.innerText = 'Tie';
        }
        announcer.classList.remove('hide');
    };

    const isValidAction = (tile) => {
        if (tile.innerText === '1' || tile.innerText === '0'){
            return false;
        }

        return true;
    };

    const updateBoard =  (index) => {
        board[index] = currentPlayer;
    }

    const changePlayer = () => {
```

```
        playerDisplay.classList.remove(`player${currentPlayer}`);
        currentPlayer = currentPlayer === '1' ? '0' : '1';
        playerDisplay.innerText = currentPlayer;
        playerDisplay.classList.add(`player${currentPlayer}`);
    }

    const userAction = (tile, index) => {
        if(isValidAction(tile) && isGameActive) {
            tile.innerText = currentPlayer;
            tile.classList.add(`player${currentPlayer}`);
            updateBoard(index);
            handleResultValidation();
            changePlayer();
        }
    }

    const resetBoard = () => {
        board = ['', '', '', '', '', '', '', '', ''];
        isGameActive = true;
        announcer.classList.add('hide');

        if (currentPlayer === '0') {
            changePlayer();
        }

        tiles.forEach(tile => {
            tile.innerText = '';
            tile.classList.remove('player1');
            tile.classList.remove('player0');
        });
    }

    tiles.forEach( (tile, index) => {
        tile.addEventListener('click', () => userAction(tile, index));
    });

    resetButton.addEventListener('click', resetBoard);
});
```

Output:


Tic-Tac-Toe
File | C:/Users/sudha_g2tyr1v/OneDrive/Desktop/captain/index.html
Gmail
YouTube
Maps
New Tab
Copy of Presentatio...
Tic Tac Toe
Player 0's turn
1 1 1
0 0
Player 1 Won
Reset

THANK YOU