```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
    <title>XOX Game</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            min-height: 100vh;
            background-color: #f5f5f5;
        }
```

```
.header {
    padding: 15px;
    text-align: center;
    background-color: #2c3e50;
    color: white;
}

.game-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    flex-grow: 1;
    padding: 20px;
}

.game-modes {
    display: flex;
    margin-bottom: 20px;
```

```
}

.mode {
    padding: 8px 16px;
    margin: 0 10px;
    cursor: pointer;
    border-radius: 4px;
    background-color: #ecf0f1;
}

.mode.active {
    background-color: #3498db;
    color: white;
}

.turn-indicator {
    font-size: 24px;
    margin: 20px 0;
    font-weight: bold;
```

```
        }

        .game-board {
            display: grid;
            grid-template-columns:
repeat(3, 100px);
            grid-template-rows: repeat(3,
100px);
            gap: 5px;
            margin-bottom: 20px;
        }

        .cell {
            width: 100px;
            height: 100px;
            background-color: #ecf0f1;
            display: flex;
            justify-content: center;
            align-items: center;
```

```
            font-size: 48px;
            font-weight: bold;
            cursor: pointer;
            border-radius: 5px;
            transition: background-color
0.2s;
        }

        .cell:hover {
            background-color: #d6eaf8;
        }

        .cell.X {
            color: #e74c3c;
        }

        .cell.O {
            color: #2ecc71;
        }
```

```
.controls {
    margin-top: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.restart-btn {
    padding: 10px 20px;
    background-color: #3498db;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
    margin-bottom: 20px;
}
```

```
        .restart-btn:hover {
            background-color: #2980b9;
        }

        .footer {
            text-align: center;
            padding: 15px;
            background-color: #2c3e50;
            color: white;
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>XOX Game</h1>
    </div>

    <div class="game-container">
        <div class="game-modes">
```

```
            <div class="mode
active">Easy</div>
            <div class="mode">Medium</
div>
            <div class="mode">Hard</div>
        </div>

        <div class="turn-indicator"
id="turnIndicator">X's Turn</div>

        <div class="game-board"
id="gameBoard">
            <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="controls">
            <button class="restart-btn"
id="restartBtn">Restart Game</
button>
        </div>
    </div>
```

```
    <div class="footer">
        <p>© 2023 XOX Game | All Rights Reserved</p>
    </div>

    <script>

document.addEventListener('DOMContentLoaded', () => {
            const cells = document.querySelectorAll('.cell');
            const turnIndicator = document.getElementById('turnIndicator');
            const restartBtn = document.getElementById('restartBtn');
            const modes =
```

```
document.querySelectorAll('.mode');

        let currentPlayer = 'X';
        let gameBoard = ['', '', '', '', '', '', '', '',
''];
        let gameActive = true;
        let gameMode = 'easy';

        // Winning combinations
        const winningCombinations = [
            [0, 1, 2], [3, 4, 5], [6, 7, 8], //
rows
            [0, 3, 6], [1, 4, 7], [2, 5, 8], //
columns
            [0, 4, 8], [2, 4, 6]             //
diagonals
        ];

        // Handle cell click
```

```
        cells.forEach(cell => {
            cell.addEventListener('click',
() => {
                const index =
cell.getAttribute('data-index');

                if (gameBoard[index] !== ''
|| !gameActive) return;

                // Make player move
                makeMove(index,
currentPlayer);

                // Check for game end
                if
(checkWin(currentPlayer)) {

turnIndicator.textContent = `${
currentPlayer} Wins!`;
```

```
                gameActive = false;
                return;
            }

            if (checkDraw()) {

turnIndicator.textContent = 'Draw!';
                gameActive = false;
                return;
            }

            // Switch player
            currentPlayer =
currentPlayer === 'X' ? 'O' : 'X';
            turnIndicator.textContent =
`${currentPlayer}'s Turn`;

            // If it's computer's turn
and in medium/hard mode
```

```
                if (gameMode !== 'easy' &&
currentPlayer === 'O') {
                    setTimeout(() => {
                        makeComputerMove();
                    }, 500);
                }
            });
        });

        // Make a move
        function makeMove(index,
player) {
            gameBoard[index] = player;
            const cell =
document.querySelector(`.cell[data-
index="${index}"]`);
            cell.textContent = player;
            cell.classList.add(player);
        }
```

```
        // Computer move logic
        function makeComputerMove()
{
            let move;

            if (gameMode === 'hard') {
                // Try to win first
                move =
findWinningMove('O');
                if (move !== -1) {
                    makeMove(move, 'O');
                    if (checkWin('O')) {

turnIndicator.textContent = 'O Wins!';
                        gameActive = false;
                    }
                    currentPlayer = 'X';
```

```
turnIndicator.textContent = "X's Turn";
                    return;
                }

                // Block player's winning
move
                move =
findWinningMove('X');
                if (move !== -1) {
                    makeMove(move, 'O');
                    currentPlayer = 'X';

turnIndicator.textContent = "X's Turn";
                    return;
                }
            }

            // Medium and Hard mode -
make random move
```

```
            const emptyCells = [];
            gameBoard.forEach((cell,
index) => {
                if (cell === '')
emptyCells.push(index);
            });

            if (emptyCells.length > 0) {
                const randomIndex =
Math.floor(Math.random() *
emptyCells.length);

makeMove(emptyCells[randomIndex],
'O');

                if (checkWin('O')) {

turnIndicator.textContent = 'O Wins!';
                    gameActive = false;
```

```
                } else if (checkDraw()) {

turnIndicator.textContent = 'Draw!';
                    gameActive = false;
                } else {
                    currentPlayer = 'X';

turnIndicator.textContent = "X's Turn";
                }
            }
        }

        // Find winning move for a
player
        function
findWinningMove(player) {
            for (const combination of
winningCombinations) {
                const [a, b, c] =
```

```
combination;
                if (gameBoard[a] ===
player && gameBoard[b] === player &&
gameBoard[c] === '') return c;
                if (gameBoard[a] ===
player && gameBoard[c] === player &&
gameBoard[b] === '') return b;
                if (gameBoard[b] ===
player && gameBoard[c] === player &&
gameBoard[a] === '') return a;
            }
            return -1;
        }

        // Check for win
        function checkWin(player) {
            return
winningCombinations.some(combinati
on => {
```

```
                return
combination.every(index => {
                    return gameBoard[index]
=== player;
                });
            });
        }

        // Check for draw
        function checkDraw() {
            return gameBoard.every(cell
=> cell !== '');
        }

        // Restart game

restartBtn.addEventListener('click', ()
=> {
            resetGame();
```

```
        });

        // Game mode selection
        modes.forEach(mode => {

mode.addEventListener('click', () => {
                modes.forEach(m =>
m.classList.remove('active'));

mode.classList.add('active');
                gameMode =
mode.textContent.toLowerCase();
                resetGame();
            });
        });

        // Reset game state
        function resetGame() {
            gameBoard = ['', '', '', '', '', '', '', '', ''];
```

```
            gameActive = true;
            currentPlayer = 'X';
            turnIndicator.textContent =
"X's Turn";

            cells.forEach(cell => {
                cell.textContent = '';
                cell.classList.remove('X',
'O');
            });

            // If computer starts in
medium/hard mode
            if (gameMode !== 'easy' &&
currentPlayer === 'O') {
                setTimeout(() => {
                    makeComputerMove();
                }, 500);
            }
```

```
        }
    });
  </script>
</body>
</html>
```