```
<!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>
   <style>
      * {
         box-sizing: border-box;
         margin: 0;
         padding: 0;
         font-family: 'Arial', sans-serif;
      }

      body {
         display: flex;
         flex-direction: column;
         align-items: center;
         justify-content: center;
         min-height: 100vh;
         background-color: #f5f5f5;
         padding: 20px;
      }

      .game-container {
         width: 100%;
         max-width: 400px;
         background-color: white;
         border-radius: 10px;
         box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
         padding: 20px;
         text-align: center;
      }

      h1 {
         color: #333;
         margin-bottom: 20px;
      }

      .game-options {
         margin-bottom: 20px;
         display: flex;
         justify-content: center;
         gap: 10px;
      }

      button {
         background-color: #4CAF50;
```

```
    color: white;
    border: none;
    padding: 10px 15px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s;
}

button:hover {
    background-color: #45a049;
}

button:disabled {
    background-color: #cccccc;
    cursor: not-allowed;
}

.game-board {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: repeat(3, 1fr);
    gap: 10px;
    margin: 20px 0;
}

.cell {
    aspect-ratio: 1/1;
    background-color: #eee;
    border-radius: 5px;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 48px;
    cursor: pointer;
    transition: background-color 0.3s;
}

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

.cell.x {
    color: #f44336;
}

.cell.o {
    color: #2196F3;
```

```
        }

        .game-info {
            margin-top: 20px;
            font-size: 18px;
            font-weight: bold;
        }

        .mobile-controls {
            display: none;
            grid-template-columns: repeat(3, 1fr);
            gap: 10px;
            margin-top: 20px;
        }

        .mobile-btn {
            padding: 15px;
            font-size: 20px;
        }

        @media (max-width: 600px) {
            .mobile-controls {
                display: grid;
            }
        }

        .restart-btn {
            margin-top: 20px;
            background-color: #ff9800;
        }

        .restart-btn:hover {
            background-color: #e68a00;
        }
    </style>
</head>
<body>
    <div class="game-container">
        <h1>Tic Tac Toe</h1>

        <div class="game-options">
            <button id="single-player-btn">Single Player</button>
            <button id="two-player-btn">Two Players</button>
        </div>

        <div class="game-board" id="board">
            <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="mobile-controls" id="mobile-controls">
            <button class="mobile-btn" data-move="0">1</button>
            <button class="mobile-btn" data-move="1">2</button>
            <button class="mobile-btn" data-move="2">3</button>
            <button class="mobile-btn" data-move="3">4</button>
            <button class="mobile-btn" data-move="4">5</button>
            <button class="mobile-btn" data-move="5">6</button>
            <button class="mobile-btn" data-move="6">7</button>
            <button class="mobile-btn" data-move="7">8</button>
            <button class="mobile-btn" data-move="8">9</button>
        </div>

        <div class="game-info" id="game-info">Select game mode to start</div>

        <button class="restart-btn" id="restart-btn">Restart Game</button>
    </div>

    <script>
        // Game variables
        let board = ['', '', '', '', '', '', '', '', ''];
        let currentPlayer = 'X';
        let gameActive = false;
        let gameMode = null;
        const winningConditions = [
            [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
        ];

        // DOM elements
        const cells = document.querySelectorAll('.cell');
        const gameInfo = document.getElementById('game-info');
        const restartBtn = document.getElementById('restart-btn');
        const singlePlayerBtn = document.getElementById('single-player-btn');
        const twoPlayerBtn = document.getElementById('two-player-btn');
        const mobileBtns = document.querySelectorAll('.mobile-btn');

        // Event listeners
        cells.forEach(cell => cell.addEventListener('click', handleCellClick));
```

```
restartBtn.addEventListener('click', restartGame);
singlePlayerBtn.addEventListener('click', () => startGame('single'));
twoPlayerBtn.addEventListener('click', () => startGame('two'));
mobileBtns.forEach(btn => btn.addEventListener('click', handleMobileMove));

// Start game with selected mode
function startGame(mode) {
   gameMode = mode;
   gameActive = true;
   currentPlayer = 'X';
   board = ['', '', '', '', '', '', '', '', ''];

   updateBoard();
   updateGameInfo(`Player ${currentPlayer}'s turn`);

   // Highlight active buttons
   singlePlayerBtn.disabled = mode === 'single';
   twoPlayerBtn.disabled = mode === 'two';
}

// Handle cell click
function handleCellClick(e) {
   const clickedCell = e.target;
   const clickedCellIndex = parseInt(clickedCell.getAttribute('data-index'));

   if (board[clickedCellIndex] !== '' || !gameActive) {
      return;
   }

   makeMove(clickedCellIndex);
}

// Handle mobile button click
function handleMobileMove(e) {
   const moveIndex = parseInt(e.target.getAttribute('data-move'));

   if (board[moveIndex] !== '' || !gameActive) {
      return;
   }

   makeMove(moveIndex);
}

// Make a move
function makeMove(index) {
   board[index] = currentPlayer;
   updateBoard();
```

```
            if (checkWin()) {
                gameActive = false;
                if (gameMode === 'single' && currentPlayer === 'O') {
                    updateGameInfo('AI wins!');
                } else {
                    updateGameInfo(`Player ${currentPlayer} wins!`);
                }
                return;
            }

            if (checkDraw()) {
                gameActive = false;
                updateGameInfo('Game ended in a draw!');
                return;
            }

            currentPlayer = currentPlayer === 'X' ? 'O' : 'X';
            updateGameInfo(`Player ${currentPlayer}'s turn`);

            // AI move in single player mode
            if (gameMode === 'single' && currentPlayer === 'O' && gameActive) {
                setTimeout(makeAIMove, 500);
            }
        }

        // AI move logic
        function makeAIMove() {
            let availableSpots = board.map((val, idx) => val === '' ? idx : null).filter(val => val !==
null);

            // Check for winning move or block opponent
            for (let player of ['O', 'X']) {
                for (let spot of availableSpots) {
                    let tempBoard = [...board];
                    tempBoard[spot] = player;
                    if (checkWin(tempBoard)) {
                        makeMove(spot);
                        return;
                    }
                }
            }

            // Try to take center
            if (board[4] === '') {
                makeMove(4);
                return;
            }
            // Take a random available spot
            const randomSpot = availableSpots[Math.floor(Math.random() *
availableSpots.length)];
            makeMove(randomSpot);
        }

        // Check for win
        function checkWin(currentBoard = board) {
            for (let condition of winningConditions) {
                const [a, b, c] = condition;
                if (currentBoard[a] && currentBoard[a] === currentBoard[b] && currentBoard[a]
=== currentBoard[c]) {
                    return true;
                }
            }
            return false;
        }

        // Check for draw
        function checkDraw() {
            return !board.includes('');
        }

        // Update board display
        function updateBoard() {
            cells.forEach((cell, index) => {
                cell.textContent = board[index];
                if (board[index] === 'X') {
                    cell.classList.add('x');
                    cell.classList.remove('o');
                } else if (board[index] === 'O') {
                    cell.classList.add('o');
                    cell.classList.remove('x');
                } else {
                    cell.classList.remove('x', 'o');
                }
            });
        }

        // Update game info text
        function updateGameInfo(message) {
            gameInfo.textContent = message;
        }

        // Restart game
        function restartGame() {
            if (gameMode) {
                startGame(gameMode);
            } else {
                gameActive = false;
                board = ['', '', '', '', '', '', '', '', ''];
                currentPlayer = 'X';
                updateBoard();
                updateGameInfo('Select game mode to start');
                singlePlayerBtn.disabled = false;
                twoPlayerBtn.disabled = false;
            }
        }
    </script>
</body>
</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>
    <style>
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
            font-family: 'Arial', sans-serif;
        }

        body {
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
            background-color: #f5f5f5;
            padding: 20px;
        }

        .game-container {
            width: 100%;
            max-width: 400px;
            background-color: white;
            border-radius: 10px;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
            padding: 20px;
            text-align: center;
        }

        h1 {
            color: #333;
            margin-bottom: 20px;
        }

        .game-options {
            margin-bottom: 20px;
            display: flex;
            justify-content: center;
            gap: 10px;
        }

        button {
            background-color: #4CAF50;
            color: white;
            border: none;
            padding: 10px 15px;
            border-radius: 5px;
            cursor: pointer;
            font-size: 16px;
            transition: background-color 0.3s;
        }

        button:hover {
            background-color: #45a049;
        }

        button:disabled {
            background-color: #cccccc;
            cursor: not-allowed;
        }

        .game-board {
            display: grid;
            grid-template-columns: repeat(3, 1fr);
            grid-template-rows: repeat(3, 1fr);
            gap: 10px;
            margin: 20px 0;
        }

        .cell {
            aspect-ratio: 1/1;
            background-color: #eee;
            border-radius: 5px;
            display: flex;
            align-items: center;
            justify-content: center;
            font-size: 48px;
            cursor: pointer;
            transition: background-color 0.3s;
```

```
        }

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

        .cell.x {
            color: #f44336;
        }

        .cell.o {
            color: #2196F3;
        }

        .game-info {
            margin-top: 20px;
            font-size: 18px;
            font-weight: bold;
        }

        .mobile-controls {
            display: none;
            grid-template-columns: repeat(3, 1fr);
            gap: 10px;
            margin-top: 20px;
        }

        .mobile-btn {
            padding: 15px;
            font-size: 20px;
        }

        @media (max-width: 600px) {
            .mobile-controls {
                display: grid;
            }
        }

        .restart-btn {
            margin-top: 20px;
            background-color: #ff9800;
        }

        .restart-btn:hover {
            background-color: #e68a00;
        }
    </style>
</head>
```

```
<body>
  <div class="game-container">
    <h1>Tic Tac Toe</h1>

    <div class="game-options">
      <button id="single-player-btn">Single Player</button>
      <button id="two-player-btn">Two Players</button>
    </div>

    <div class="game-board" id="board">
      <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="mobile-controls" id="mobile-controls">
      <button class="mobile-btn" data-move="0">1</button>
      <button class="mobile-btn" data-move="1">2</button>
      <button class="mobile-btn" data-move="2">3</button>
      <button class="mobile-btn" data-move="3">4</button>
      <button class="mobile-btn" data-move="4">5</button>
      <button class="mobile-btn" data-move="5">6</button>
      <button class="mobile-btn" data-move="6">7</button>
      <button class="mobile-btn" data-move="7">8</button>
      <button class="mobile-btn" data-move="8">9</button>
    </div>

    <div class="game-info" id="game-info">Select game mode to start</div>

    <button class="restart-btn" id="restart-btn">Restart Game</button>
  </div>

  <script>
    // Game variables
    let board = ['', '', '', '', '', '', '', '', ''];
    let currentPlayer = 'X';
    let gameActive = false;
    let gameMode = null;
    const winningConditions = [
      [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
```

```
];

// DOM elements
const cells = document.querySelectorAll('.cell');
const gameInfo = document.getElementById('game-info');
const restartBtn = document.getElementById('restart-btn');
const singlePlayerBtn = document.getElementById('single-player-btn');
const twoPlayerBtn = document.getElementById('two-player-btn');
const mobileBtns = document.querySelectorAll('.mobile-btn');

// Event listeners
cells.forEach(cell => cell.addEventListener('click', handleCellClick));
restartBtn.addEventListener('click', restartGame);
singlePlayerBtn.addEventListener('click', () => startGame('single'));
twoPlayerBtn.addEventListener('click', () => startGame('two'));
mobileBtns.forEach(btn => btn.addEventListener('click', handleMobileMove));

// Start game with selected mode
function startGame(mode) {
   gameMode = mode;
   gameActive = true;
   currentPlayer = 'X';
   board = ['', '', '', '', '', '', '', '', ''];

   updateBoard();
   updateGameInfo(`Player ${currentPlayer}'s turn`);

   // Highlight active buttons
   singlePlayerBtn.disabled = mode === 'single';
   twoPlayerBtn.disabled = mode === 'two';
}

// Handle cell click
function handleCellClick(e) {
   const clickedCell = e.target;
   const clickedCellIndex = parseInt(clickedCell.getAttribute('data-index'));

   if (board[clickedCellIndex] !== '' || !gameActive) {
      return;
   }

   makeMove(clickedCellIndex);
}

// Handle mobile button click
function handleMobileMove(e) {
   const moveIndex = parseInt(e.target.getAttribute('data-move'));
```

```
        if (board[moveIndex] !== '' || !gameActive) {
            return;
        }

        makeMove(moveIndex);
    }

    // Make a move
    function makeMove(index) {
        board[index] = currentPlayer;
        updateBoard();

        if (checkWin()) {
            gameActive = false;
            if (gameMode === 'single' && currentPlayer === 'O') {
                updateGameInfo('AI wins!');
            } else {
                updateGameInfo(`Player ${currentPlayer} wins!`);
            }
            return;
        }

        if (checkDraw()) {
            gameActive = false;
            updateGameInfo('Game ended in a draw!');
            return;
        }

        currentPlayer = currentPlayer === 'X' ? 'O' : 'X';
        updateGameInfo(`Player ${currentPlayer}'s turn`);

        // AI move in single player mode
        if (gameMode === 'single' && currentPlayer === 'O' && gameActive) {
            setTimeout(makeAIMove, 500);
        }
    }

    // AI move logic
    function makeAIMove() {
        let availableSpots = board.map((val, idx) => val === '' ? idx : null).filter(val => val !==
null);

        // Check for winning move or block opponent
        for (let player of ['O', 'X']) {
            for (let spot of availableSpots) {
                let tempBoard = [...board];
                tempBoard[spot] = player;
                if (checkWin(tempBoard)) {
                        makeMove(spot);
                        return;
                    }
                }
            }

            // Try to take center
            if (board[4] === '') {
                makeMove(4);
                return;
            }

            // Take a random available spot
            const randomSpot = availableSpots[Math.floor(Math.random() *
availableSpots.length)];
            makeMove(randomSpot);
        }

        // Check for win
        function checkWin(currentBoard = board) {
            for (let condition of winningConditions) {
                const [a, b, c] = condition;
                if (currentBoard[a] && currentBoard[a] === currentBoard[b] && currentBoard[a]
=== currentBoard[c]) {
                    return true;
                }
            }
            return false;
        }

        // Check for draw
        function checkDraw() {
            return !board.includes('');
        }

        // Update board display
        function updateBoard() {
            cells.forEach((cell, index) => {
                cell.textContent = board[index];
                if (board[index] === 'X') {
                    cell.classList.add('x');
                    cell.classList.remove('o');
                } else if (board[index] === 'O') {
                    cell.classList.add('o');
                    cell.classList.remove('x');
                } else {
                    cell.classList.remove('x', 'o');
                }
            });
        }

        // Update game info text
        function updateGameInfo(message) {
            gameInfo.textContent = message;
        }

        // Restart game
        function restartGame() {
            if (gameMode) {
                startGame(gameMode);
            } else {
                gameActive = false;
                board = ['', '', '', '', '', '', '', '', ''];
                currentPlayer = 'X';
                updateBoard();
                updateGameInfo('Select game mode to start');
                singlePlayerBtn.disabled = false;
                twoPlayerBtn.disabled = false;
            }
        }
    </script>
</body>
</html>
```