KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai, Erode – 638 060

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

GAMING VERSE

AN APPLICATION PROJECT REPORT

for

<u>22ADC21-DATA STRUCTURES</u>

SUBMITED BY:

SIVA DHARSHANA.G(23ADR151)

SUBIKSHA.A(23ADR162)

VARUNIKA.EBB(23ADR183)

CODING FOR MAIN WEBSITE PAGE:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>GamingVerse</title>
<link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;70
0&family=Lobster&display=swap" rel="stylesheet">
<link rel="stylesheet" href="game.css">
</head>
<body>
<nav class="navbar">
<div class="container">
<a class="navbar-brand" href="#">GamingVerse</a>
</div>
</nav>
<header class="hero">
<div class="container">
<h1 class="hero-title">Immersive Gaming Experiences</h1>
<p class="hero-subtitle"></p>
</div>
</header>
<main class="container">
<section id="about" class="about-section">
<h2>About US</h2>
<div class="about-content">
<img src="game.jpeg" alt="Your Photo" class="about-photo">
```

```html
<b>We bring classic games to life: Tic-Tac-Toe, a strategy game
aligning Xs and Os; 4 in a Row, where players connect discs in a grid;
and the Matching Images Game, a memory challenge matching
identical cards.</b>
</div>
</section>
<section id="work" class="work-section">
<h2>Games</h2>
<div class="work-cards">
<div class="work-card">
<img src="tic-tac-toe.jpg" alt="Logo 1" height="300">
<div class="work-card-content">
<h3>TIC-TAC-TOE</h3>
<p>The classic game of Xs and Os where strategy meets
simplicity.</p><br>
<a href="index.html" class="btn btn-primary">Play Now</a>
</div>
</div>
<div class="work-card">
<img src="4inarow.jpg" alt="Logo 2" height="300">
<div class="work-card-content">
<h3>4 In A Row </h3>
<p>"Connect four of your colored discs in a row to win in this classic
strategy game of 4 in a Row."</p>
            <a href="index1.html" class="btn btn-primary">Play Now</a>
</div>
</div>
<div class="work-card">
<img src="memorygame.jpg" alt="Logo 3" height="300">
<div class="work-card-content">
<h3>Memory Game</h3>
<p>Flip cards to find pairs and challenge your memory!</p><br>
```

```
<a href="index2.html" class="btn btn-primary">Play Now</a>
</div>
</div>
</div>
</section>
</main>
<footer class="footer">
<div class="container">
<p>&copy; 2024 Gaming world</p>
</div>
</footer>
<script src="game1.js"></script>
</body>
</html>
```

HTML CODING FOR TIC-TAC-TOE GAME:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>4 in a Row</title>
   <link rel="stylesheet" href="styles1.css">
</head>
<body>
   <div id="game">
      <h1>4 in a Row</h1>
      <div id="board">
```

```
            <!-- Cells will be generated by JavaScript -->
        </div>
        <button id="reset">Reset Game</button>
        <div id="message"></div>
    </div>
    <script src="script1.js"></script>
</body>
</html>
```

JS CODING FOR TIC-TAC-TOE GAME:

```
document.addEventListener("DOMContentLoaded", () => {
    class MoveNode {
        constructor(index, player) {
            this.index = index;
            this.player = player;
            this.next = null;
        }
    }

    class MoveList {
        constructor() {
            this.head = null;
        }

        addMove(index, player) {
            const newNode = new MoveNode(index, player);
            if (!this.head) {
                this.head = newNode;
            } else {
                let current = this.head;
                while (current.next) {
```

```
                current = current.next;
            }
            current.next = newNode;
        }
    }

    reset() {
        this.head = null;
    }
}

class GameStateNode {
    constructor(board) {
        this.board = board.slice();
        this.left = null;
        this.right = null;
    }
}

class GraphNode {
    constructor(index) {
        this.index = index;
        this.adjacent = [];
    }

    addAdjacent(node) {
        this.adjacent.push(node);
    }
}

function createWinningGraph() {
        const nodes = Array.from({ length: 9 }, (_, i) => new
GraphNode(i));
        const winningPatterns = [
            [0, 1, 2], [3, 4, 5], [6, 7, 8],
            [0, 3, 6], [1, 4, 7], [2, 5, 8],
            [0, 4, 8], [2, 4, 6]
        ];

        for (const pattern of winningPatterns) {
            const [a, b, c] = pattern;
            nodes[a].addAdjacent(nodes[b]);
            nodes[a].addAdjacent(nodes[c]);
            nodes[b].addAdjacent(nodes[a]);
            nodes[b].addAdjacent(nodes[c]);
            nodes[c].addAdjacent(nodes[a]);
            nodes[c].addAdjacent(nodes[b]);
        }

        return nodes;
    }

    const board = document.querySelector("#board");
    const cells = document.querySelectorAll(".cell");
    const resetButton = document.querySelector("#reset");
    const message = document.querySelector("#message");
    let currentPlayer = "X";
    let gameState = new GameStateNode(Array(9).fill(""));
    let moves = new MoveList();
    const winningGraph = createWinningGraph();

    function handleCellClick(event) {
        const cell = event.target;
        const cellIndex = parseInt(cell.getAttribute("data-index"));

        if (gameState.board[cellIndex] !== "" || checkWinner()) {
            return;
        }

        gameState.board[cellIndex] = currentPlayer;
        moves.addMove(cellIndex, currentPlayer);
        cell.textContent = currentPlayer;

        if (checkWinner()) {
            message.textContent = `${currentPlayer} wins!`;
        } else if (gameState.board.every(cell => cell !== "")) {
            message.textContent = "It's a draw!";
        } else {
            currentPlayer = currentPlayer === "X" ? "O" : "X";
        }
    }

    function checkWinner() {
        const board = gameState.board;
        const winningPatterns = [
            [0, 1, 2], [3, 4, 5], [6, 7, 8],
            [0, 3, 6], [1, 4, 7], [2, 5, 8],
            [0, 4, 8], [2, 4, 6]
        ];

        for (const pattern of winningPatterns) {
            const [a, b, c] = pattern;
            if (board[a] && board[a] === board[b] && board[a] ===
board[c]) {
                return true;
            }
        }
        return false;
    }

    function resetGame() {
        gameState = new GameStateNode(Array(9).fill(""));
        moves.reset();
        currentPlayer = "X";
        message.textContent = "";
        cells.forEach(cell => {
            cell.textContent = "";
        });
    }

    cells.forEach(cell => {
        cell.addEventListener("click", handleCellClick);
    });

    resetButton.addEventListener("click", resetGame);
});
```

CSS CODING FOR TIC-TAC-TOE GAME:

```
body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-image: url("img4.jpg");
    margin: 0;
    font-family: Arial, sans-serif;
    background-repeat: no-repeat;
}

#game {
    text-align: center;
}

#board {
    display: grid;
    grid-template-columns: repeat(3, 100px);
    gap: 10px;
    margin: 20px auto;
}

.cell {
    width: 100px;
    height: 100px;
    background-color: #fff;
    border: 2px solid #000;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 2rem;
    cursor: pointer;
}

.cell:hover {
    background-color: #f1f1f1;
}

#reset {
    margin-top: 20px;
```

```
        padding: 10px 20px;
        font-size: 1rem;
    }

    #message {
        margin-top: 20px;
        font-size: 1.2rem;
    }
```

HTML CODING FOR 4 IN A ROW GAME:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>4 in a Row</title>
    <link rel="stylesheet" href="styles1.css">
</head>
<body>
    <div id="game">
        <h1>4 in a Row</h1>
        <div id="board">
            <!-- Cells will be generated by JavaScript -->
        </div>
        <button id="reset">Reset Game</button>
        <div id="message"></div>
    </div>
    <script src="script1.js"></script>
</body>
</html>
```

JS CODING FOR 4 IN A ROW:

```
document.addEventListener("DOMContentLoaded", () => {
    const ROWS = 6;
    const COLS = 7;
    const CONNECT = 4;

    class MoveNode {
        constructor(row, col, player) {
            this.row = row;
            this.col = col;
            this.player = player;
            this.next = null;
        }
    }

    class MoveList {
        constructor() {
            this.head = null;
        }

        addMove(row, col, player) {
            const newNode = new MoveNode(row, col, player);
            if (!this.head) {
                this.head = newNode;
            } else {
                let current = this.head;
                while (current.next) {
                    current = current.next;
                }
                current.next = newNode;
            }
        }

        reset() {
            this.head = null;
        }
    }

    class TreeNode {
        constructor(row, col, player) {
            this.row = row;
            this.col = col;
            this.player = player;
            this.children = [];
        }

        addChild(node) {
            this.children.push(node);
        }
    }

    class GraphNode {
        constructor(row, col) {
            this.row = row;
            this.col = col;
            this.adjacent = [];
        }

        addAdjacent(node) {
            this.adjacent.push(node);
        }
    }

    function createGameGraph() {
        const graph = [];
        for (let r = 0; r < ROWS; r++) {
            graph[r] = [];
            for (let c = 0; c < COLS; c++) {
                graph[r][c] = new GraphNode(r, c);
            }
        }

        for (let r = 0; r < ROWS; r++) {
            for (let c = 0; c < COLS; c++) {
                if (r < ROWS - 1) graph[r][c].addAdjacent(graph[r + 1][c]);
                if (c < COLS - 1) graph[r][c].addAdjacent(graph[r][c + 1]);
                if (r < ROWS - 1 && c < COLS - 1)
graph[r][c].addAdjacent(graph[r + 1][c + 1]);
                if (r < ROWS - 1 && c > 0) graph[r][c].addAdjacent(graph[r
+ 1][c - 1]);
            }
        }
        return graph;
    }

    const board = document.querySelector("#board");
    const resetButton = document.querySelector("#reset");
    const message = document.querySelector("#message");
    const cells = [];

    for (let r = 0; r < ROWS; r++) {
        cells[r] = [];
        for (let c = 0; c < COLS; c++) {
            const cell = document.createElement("div");
            cell.className = "cell";
            cell.dataset.row = r;
            cell.dataset.col = c;
            board.appendChild(cell);
            cells[r][c] = cell;
        }
    }

    const moveList = new MoveList();
    let currentPlayer = "red";
    const gameGraph = createGameGraph();

    function handleCellClick(event) {
        const cell = event.target;
        const col = parseInt(cell.dataset.col);

        if (checkWinner()) return;

        let targetRow = -1;
        for (let r = ROWS - 1; r >= 0; r--) {
            if (!cells[r][col].classList.contains("red")
&& !cells[r][col].classList.contains("blue")) {
                targetRow = r;
                break;
            }
        }

        if (targetRow === -1) return;

        const playerClass = currentPlayer;
        moveList.addMove(targetRow, col, playerClass);
        cells[targetRow][col].classList.add(playerClass);

        if (checkWinner()) {
            message.textContent =
`${currentPlayer.charAt(0).toUpperCase() + currentPlayer.slice(1)}
wins!`;
        } else {
            currentPlayer = currentPlayer === "red" ? "blue" : "red";
        }
    }

    function checkWinner() {
        const directions = [
            [ [0, 1], [0, -1] ],
            [ [1, 0], [-1, 0] ],
            [ [1, 1], [-1, -1] ],
            [ [1, -1], [-1, 1] ]
        ];

        function isValid(row, col) {
            return row >= 0 && row < ROWS && col >= 0 && col <
COLS;
        }

        for (let r = 0; r < ROWS; r++) {
            for (let c = 0; c < COLS; c++) {
                if (cells[r][c].classList.contains(currentPlayer)) {
                    for (const direction of directions) {
                        let count = 1;
                        for (const [dr, dc] of direction) {
                            let nr = r + dr;
                            let nc = c + dc;
                            while (isValid(nr, nc) &&
cells[nr][nc].classList.contains(currentPlayer)) {
                            count++;
                            nr += dr;
                            nc += dc;
                            }
                        }
                        if (count >= CONNECT) return true;
                    }
                }
            }
        }
        return false;
    }

    function resetGame() {
        for (let r = 0; r < ROWS; r++) {
            for (let c = 0; c < COLS; c++) {
                cells[r][c].classList.remove("red", "blue");
            }
        }
        moveList.reset();
        message.textContent = "";
        currentPlayer = "red";
    }

    cells.forEach(row => {
        row.forEach(cell => {
            cell.addEventListener("click", handleCellClick);
        });
    });
```

```
    resetButton.addEventListener("click", resetGame);
});
```

CSS CODING FOR 4 IN A ROW:

```
body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-image: url("img4.jpg");
    margin: 0;
    font-family: Arial, sans-serif;
    background-repeat: no-repeat;

}

#game {
    text-align: center;
}

#board {
    display: grid;
    grid-template-columns: repeat(7, 50px);
    grid-gap: 5px;
    margin: 20px auto;
    justify-content: center;
}

.cell {
    width: 50px;
    height: 50px;
    background-color: #fff;
    border: 2px solid #000;
    display: flex;
    justify-content: center;
    align-items: center;
    cursor: pointer;
    transition: background-color 0.3s;
}

.cell.red {
    background-color: rgb(245, 51, 83);
}

.cell.blue {
    background-color: rgba(114, 123, 228, 0.911);
}

#reset {
    margin-top: 20px;
    padding: 10px 20px;
    font-size: 1rem;
}

#message {
    margin-top: 20px;
    font-size: 1.2rem;
}
```

HTML CODING FOR MEMORY GAME:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Memory Game</title>
   <link rel="stylesheet" href="styles2.css">
</head>
<body>
   <div class="container">
      <div class="game-container" id="game-container"></div>
   </div>
   <script src="script2.js"></script>
</body>
</html>
```

JS CODING FOR MEMORY GAME:

```
// Game logic
const cards = [];
const images = [
  "https://e7.pngegg.com/pngimages/496/496/png-clipart-dora-the-
explorer-illustration-dora-animated-cartoon-character-cartoon-
characters-dora-the-explorer-s-miscellaneous-television.png",
  "https://cn.i.cdn.ti-
platform.com/content/2302/pokemon/showpage/za/pokemon_icon_cm
s.ec3b1bb3.png",
  "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSAf22Vce6NmiByfgU9balaR
VUpp_Jfp51VWg&usqp=CAU",
  "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTyV5dFF0lhZRi5D0wMFgy
NjW2dGkT_C1JsUg&usqp=CAU",
  "https://www.animaker.com/hub/wp-
content/uploads/2023/03/Mickey_Mouse_Disney_1.webp",
  "https://m.media-
amazon.com/images/I/51DwGfBvcBL._AC_UF894,1000_QL80_.jpg",

"https://i.pinimg.com/originals/9b/a2/57/9ba25796112cad616be27e473
ae1e149.jpg",
  "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTaxkYo0mIcMOAJFrzJciMm
JA12GRdt0mlXKA&usqp=CAU"
];

let firstCardClicked = null;
let secondCardClicked = null;

function createGame() {
  for (let i = 0; i < 8; i++) {
    cards.push({ id: i, imageUrl: images[i], matched: false });
    cards.push({ id: i, imageUrl: images[i], matched: false });
  }
  shuffle(cards);
  displayCards(cards);
}

function shuffle(array) {
  for (let i = array.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [array[i], array[j]] = [array[j], array[i]];
  }
}

function displayCards(cards) {
  const gameContainer = document.getElementById("game-
container");
  cards.forEach(card => {
    const cardElement = document.createElement("div");
    cardElement.classList.add("card");
    const imgElement = document.createElement("img");
    imgElement.src =
"https://via.placeholder.com/150/000000/FFFFFF?text=Closed";
    imgElement.dataset.imageUrl = card.imageUrl;
    imgElement.addEventListener("click", () =>
handleCardClick(imgElement));
    cardElement.appendChild(imgElement);
    gameContainer.appendChild(cardElement);
  });
}

function handleCardClick(imgElement) {
  if (!firstCardClicked) {
    firstCardClicked = imgElement;
    firstCardClicked.src = firstCardClicked.dataset.imageUrl;
  } else if (!secondCardClicked) {
    secondCardClicked = imgElement;
    secondCardClicked.src = secondCardClicked.dataset.imageUrl;
    setTimeout(checkForMatch, 1000);
  }
}

function checkForMatch() {
  if (firstCardClicked.dataset.imageUrl ===
secondCardClicked.dataset.imageUrl) {
    firstCardClicked.removeEventListener("click", () =>
handleCardClick(firstCardClicked));
```

```
        secondCardClicked.removeEventListener("click", () =>
handleCardClick(secondCardClicked));
    } else {
        firstCardClicked.src =
"https://via.placeholder.com/150/000000/FFFFFF?text=Closed";
        secondCardClicked.src =
"https://via.placeholder.com/150/000000/FFFFFF?text=Closed";
    }
    firstCardClicked = null;
    secondCardClicked = null;
}

createGame();
```

CSS CODING FOR MEMORY GAME:

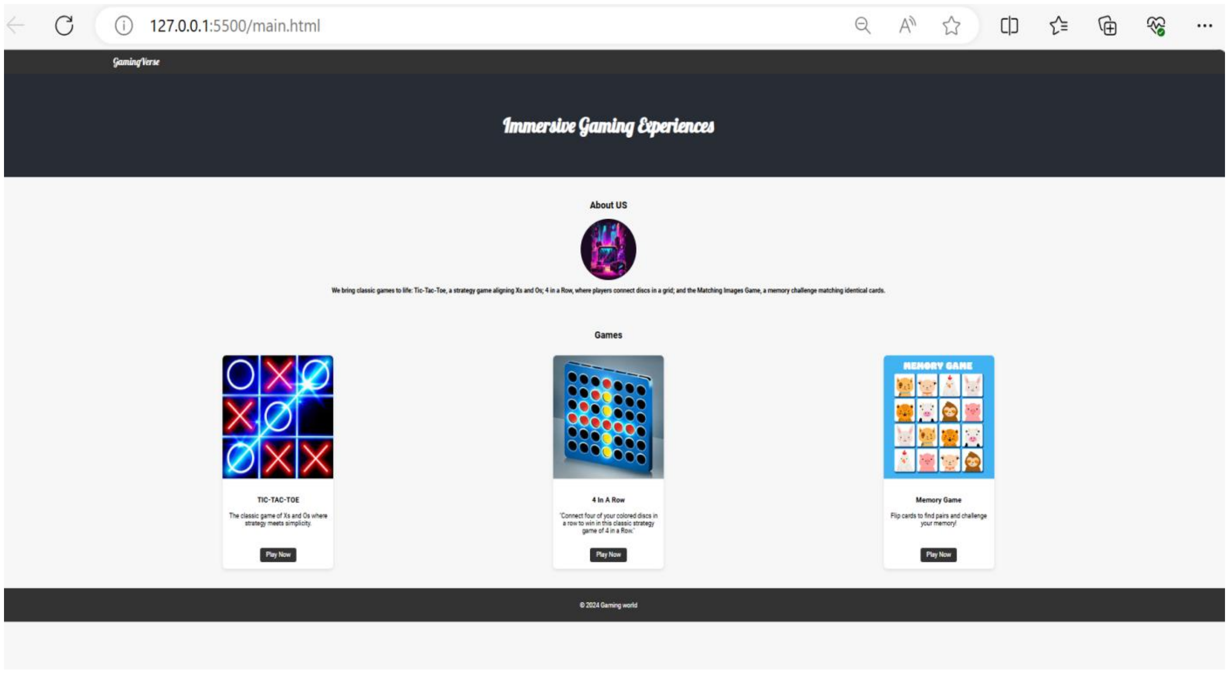
```
.container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.game-container {
    display: grid;
    grid-template-columns: repeat(4, 150px);
    grid-template-rows: repeat(4, 150px);
    gap: 5px;
}

.card {
    width: 100%;
    height: 100%;
    background-color: #a1f6f2;
    cursor: pointer;
    border: 1px solid #ccc;
    overflow: hidden;
    box-sizing: border-box;
}

.card img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}
```

OUTPUT FOR MAIN WEBSITE:



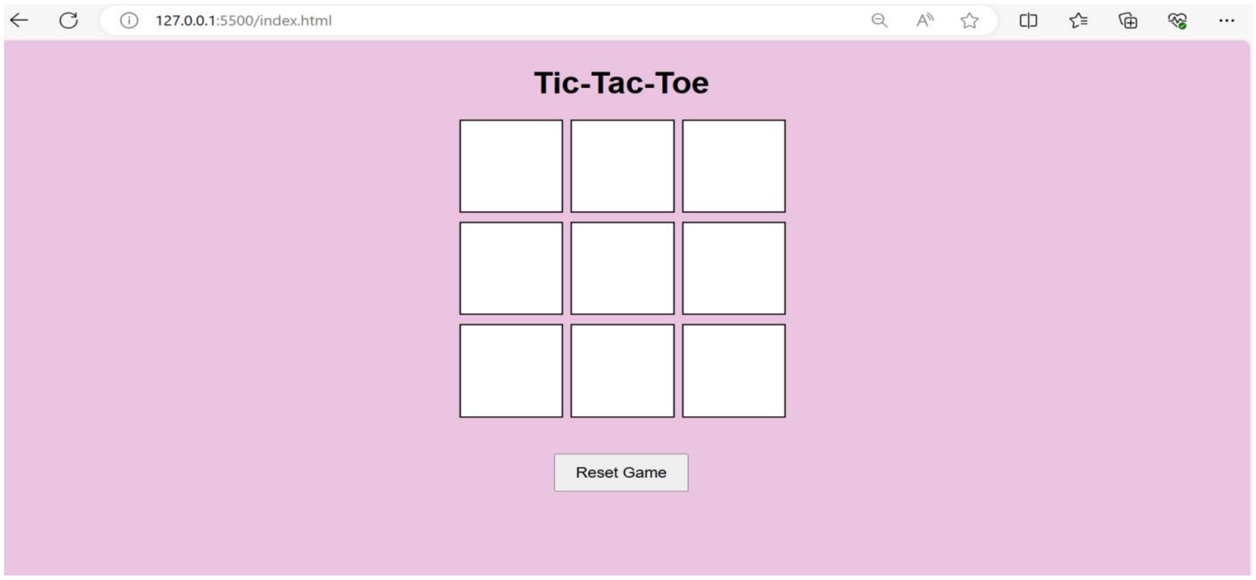
We bring classic games to life: Tic-Tac-Toe, a strategy game aligning Xs and Os; 4 in a Row, where players connect discs in a grid; and the Matching Images Game, a memory challenge matching identical cards.

The classic game of Xs and Os where strategy meets simplicity.

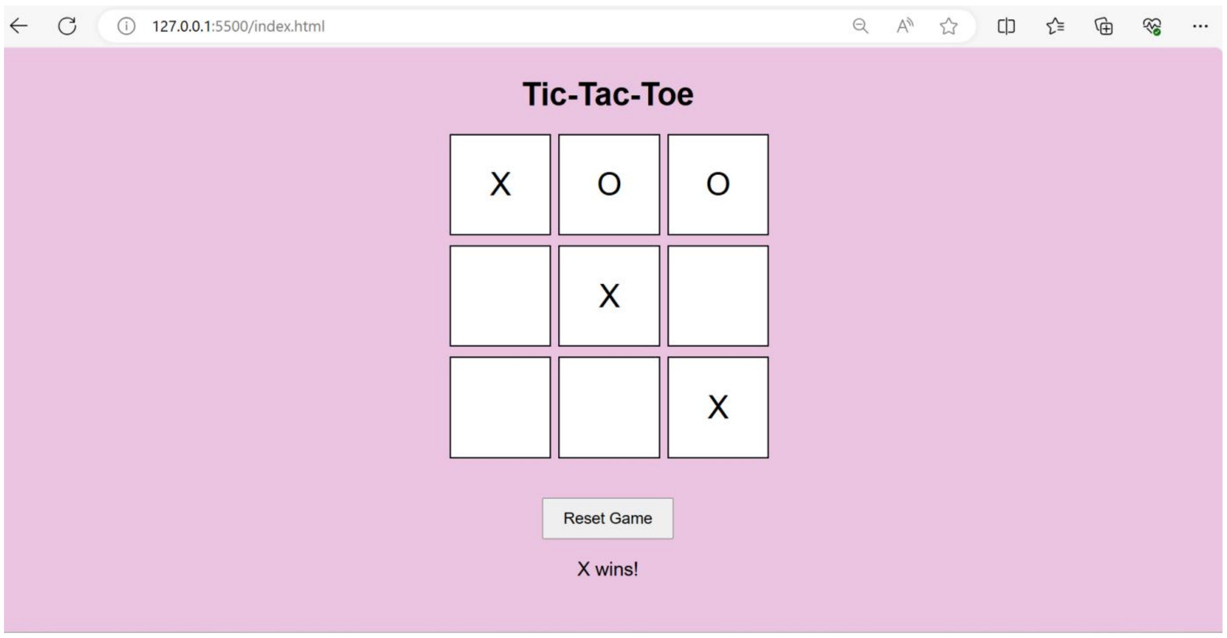
"Connect four of your colored discs in a row to win in this classic strategy game of 4 in a Row."

Flip cards to find pairs and challenge your memory!

OUTPUT FOR TIC-TAC-TOE GAME:

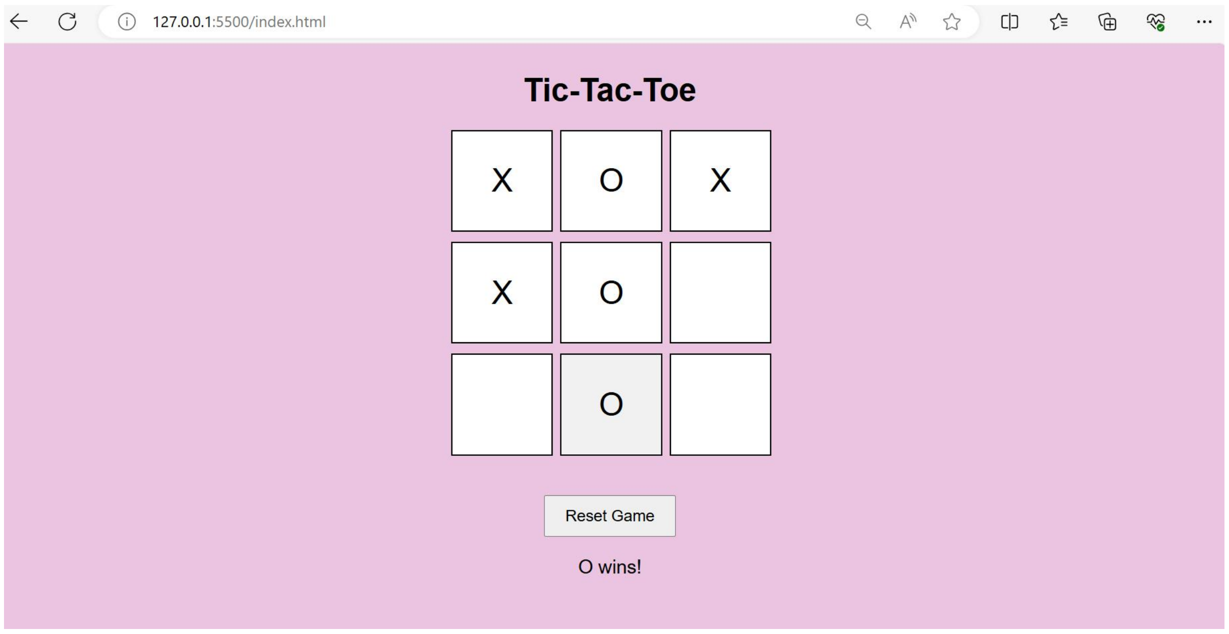


OUTPUT WHEN X WINS:



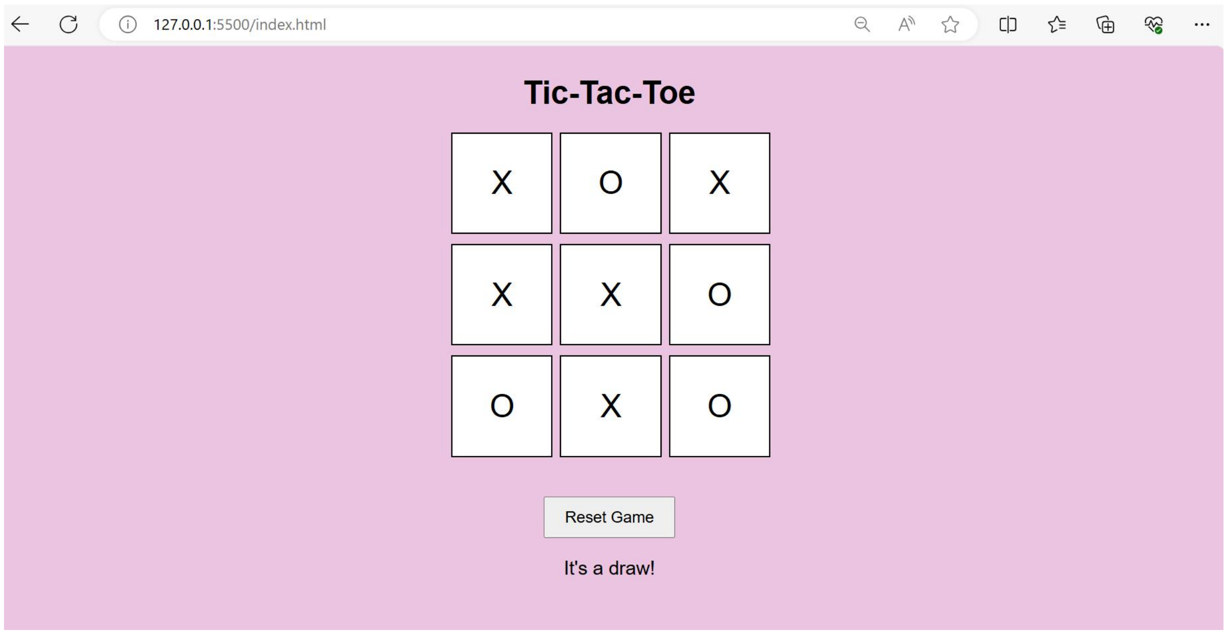
X wins!

OUTPUT WHEN O WINS:

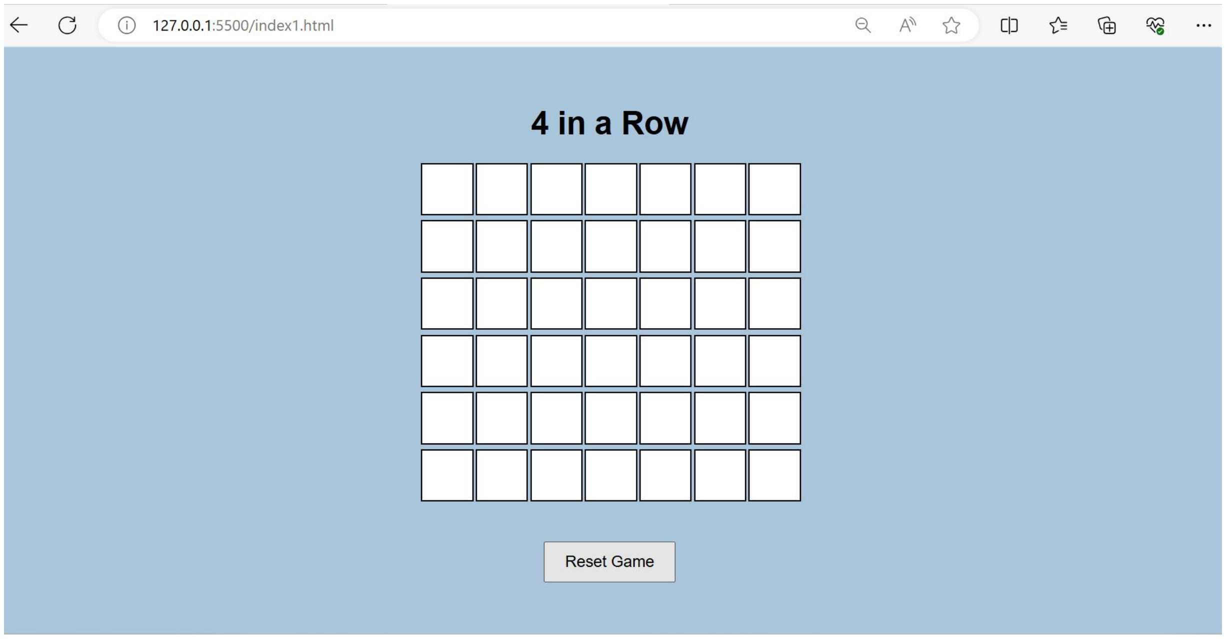


O wins!

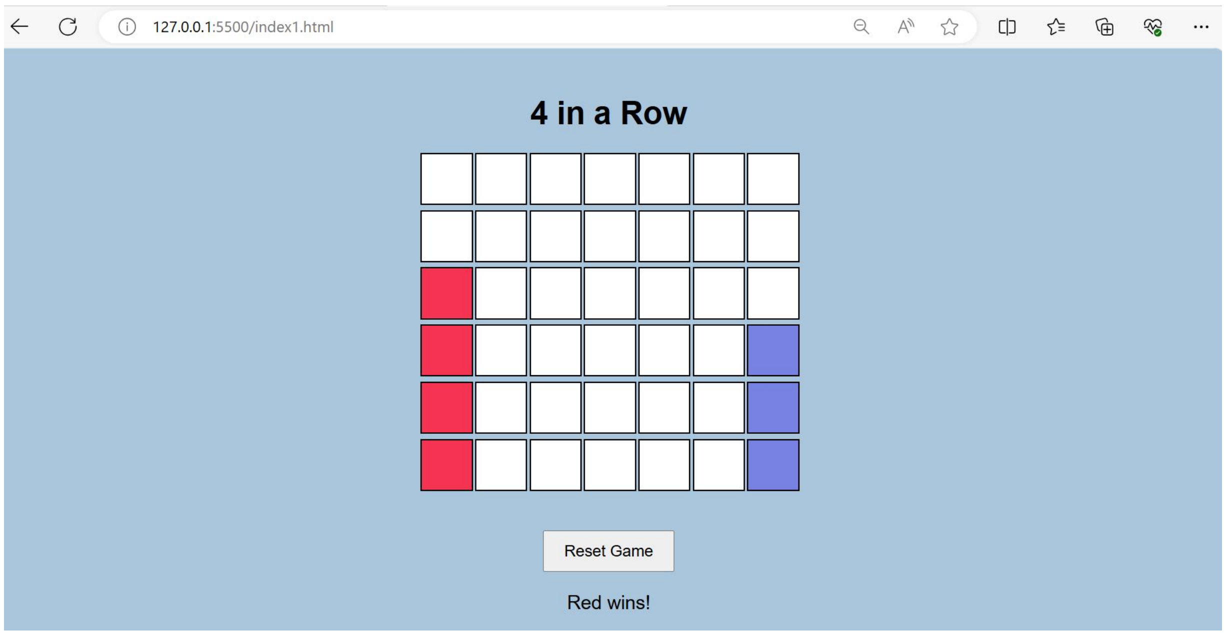
OUTPUT WHEN IT IS A DRAW:



OUTPUT FOR 4-IN-A-ROW GAME:



OUTPUT WHEN RED WINS:

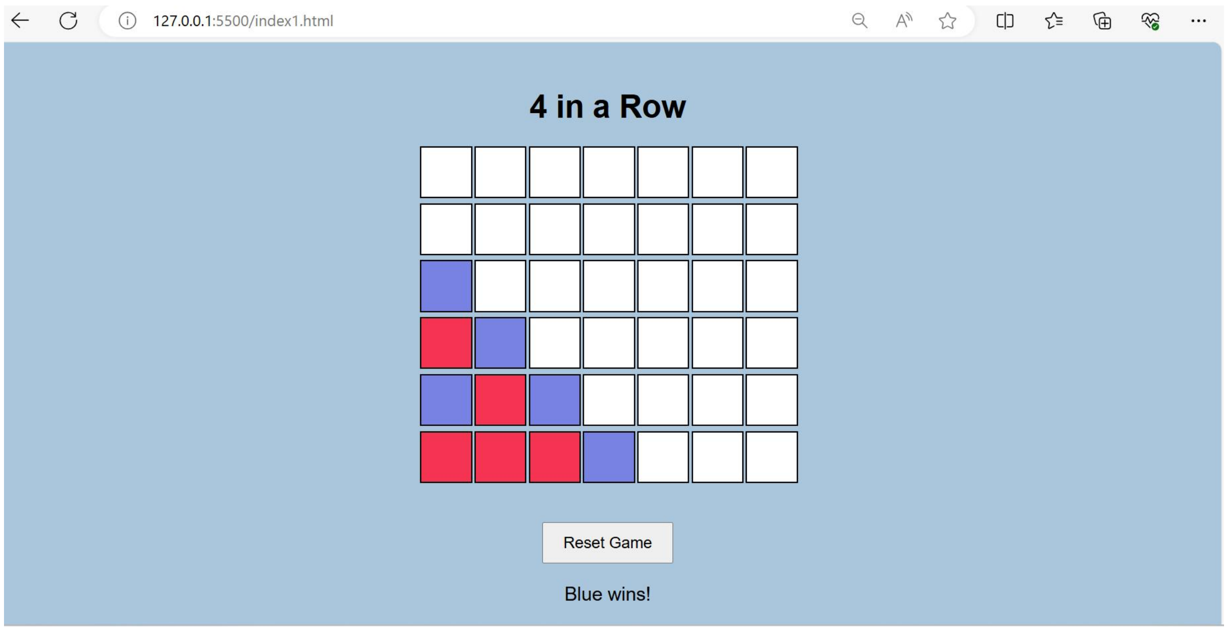

127.0.0.1:5500/index1.html

4 in a Row

Reset Game

Red wins!

OUTPUT WHEN BLUE WINS:

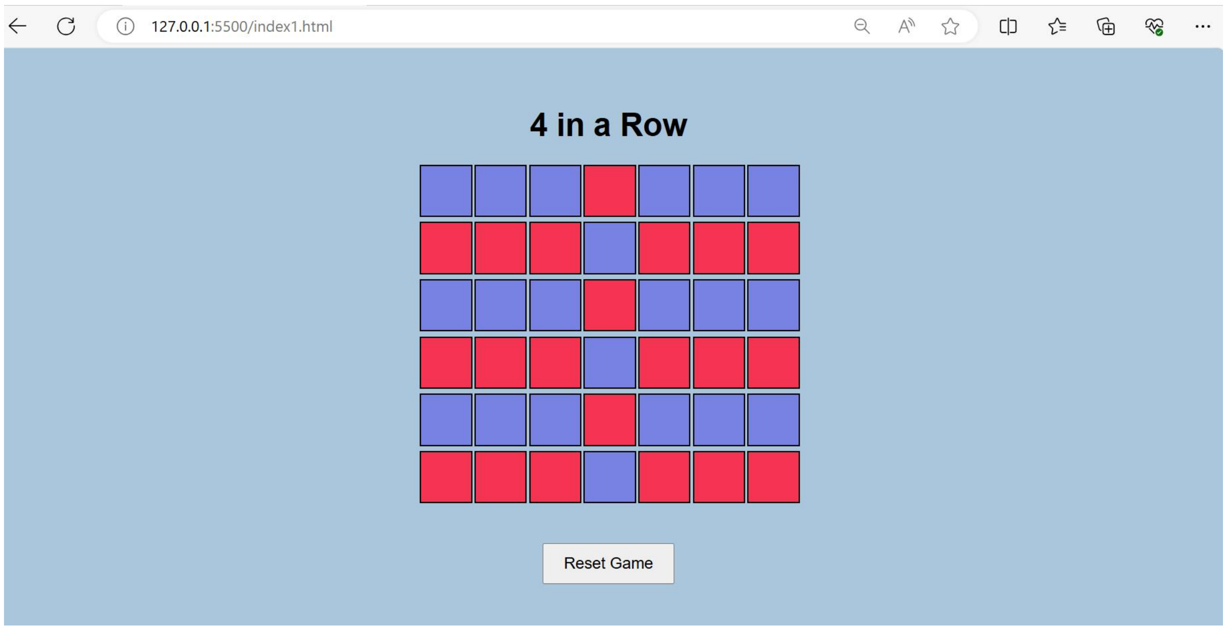

127.0.0.1:5500/index1.html

4 in a Row

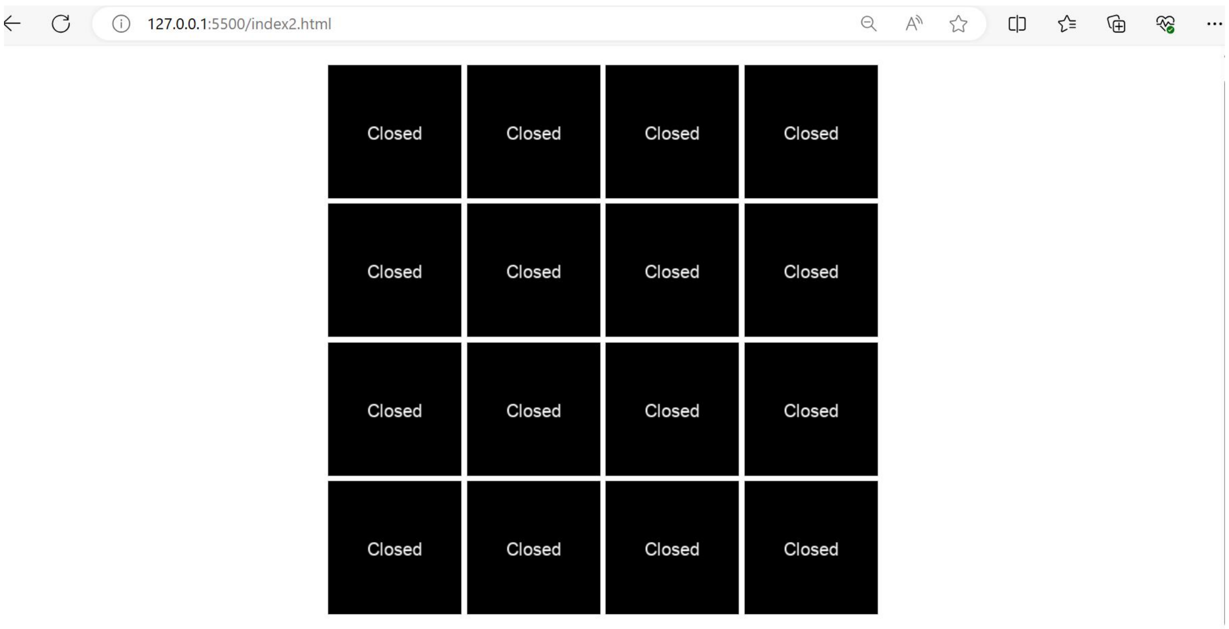
Reset Game

Blue wins!

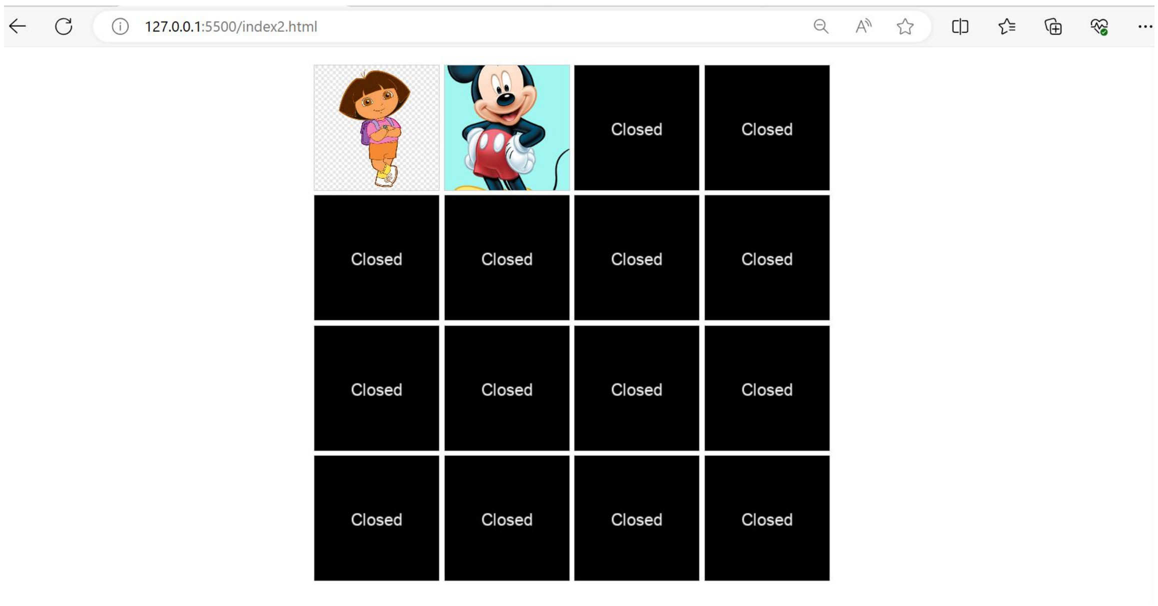
OUTPUT WHEN IT IS A DRAW:



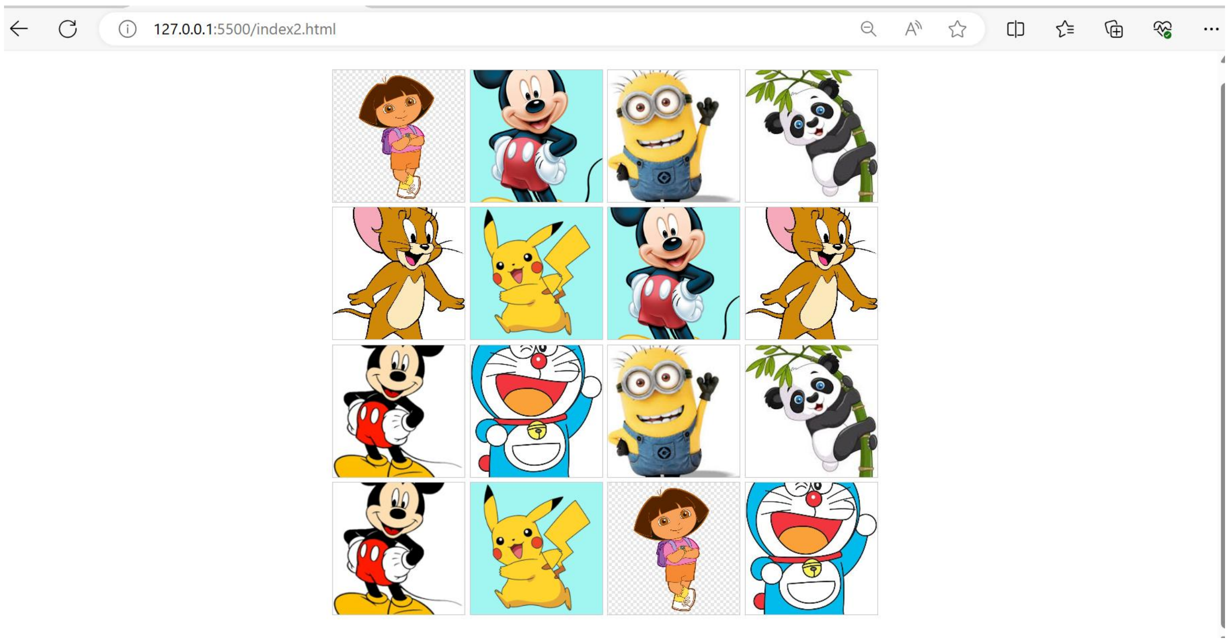
OUTPUT FOR PICTURE MEMORY GAME:



OUTPUT WHEN PICTURE MISMATCHES:



OUTPUT FOR CORRECT MATCHES:


127.0.0.1:5500/index2.html