RAMAIAH INSTITUTE OF TECHNOLOGY

MSRIT NAGAR, BENGALURU, 560054

A Report on

MEMORY MATCHING GAME

Submitted in partial fulfilment of the OTHER COMPONENT requirements as a part of the JavaScript and JQuery subject with code ISAEC493 for the IV Semester of degree of

Bachelor of Engineering in Information Science and Engineering

Submitted by

Candidate Names

PRAJWAL.K (1MS22IS096)

SHARATH.S (1MS22IS123)

Under the Guidance of

Faculty In-charge

MR.SHIVANANDA S

Assistant professor

Department of ISE

Department of Information Science and Engineering

Ramaiah Institute of Technology

2023 - 2024

Project Report: Memory Matching Game

1. Introduction

The Memory Matching Game is a classic concentration game where players match pairs of identical tiles. This project implements the game using HTML, CSS, and JavaScript to create an interactive and visually appealing web-based version.

2. Objective

The objective of this project is to develop a memory game that:

- Engages users with interactive gameplay.
- Enhances user experience through attractive design and smooth animations.
- Implements game logic to track moves, time, and game completion.
- Provides options for different game modes (grid sizes).

3. Technologies Used

- **Frontend:** HTML5, CSS3 (including CSS Grid for layout), JavaScript (ES6)
- **Libraries:** jQuery for DOM manipulation and animation
- **External Resources:** Google Fonts (Biryani), jQuery CDN for library inclusion

4. Features Implemented

4.1. User Interface (UI)

- **Header:** Displays game title ("Memory Game") and statistics (Moves, Time).
- **Game Board:** Dynamically creates a grid of tiles using <table> and <td> elements.
- **Tiles:** Each tile consists of a front and back face, flipped via CSS transitions on click.

- **Overlay:** Provides game instructions and end-game messages with options to start new games or exit.
- **Buttons:** Responsive buttons for selecting different game modes and restarting the game.

4.2. Gameplay Mechanics

- **Tile Matching:** Allows players to click tiles to reveal their content and attempts to match identical pairs.
- **Move Counter:** Tracks the number of moves taken by the player.
- **Timer:** Displays the elapsed time since starting the game.
- **Game Completion:** Detects when all pairs have been correctly matched and displays a congratulatory message with game statistics.

4.3. Design and Styling

- **Visual Appeal:** Uses gradients, box shadows, and animations (like background gradient animation) to enhance visual appeal and user engagement.
- **Responsive Design:** Ensures the game adapts to different screen sizes using CSS @media queries and viewport settings.

5. Implementation Details

5.1. HTML Structure

- Structured with semantic HTML5 elements (<header>, <main>, <table>, <div> for overlay).
- Links to external CSS (gameStyle.css) and JavaScript (gamescript.js) files.

index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="gameStyle.css">
    <title>Memory Game</title>
</head>
<body>
    <header>
        <div id="logo">Memory Game</div>
        <div id="stats">
            <div id="moves">Moves: 0</div>
            <div id="time">Time: 00:00</div>
        </div>
    </header>
    <main>
        <table></table>
    </main>
    <div id="overlay"></div>
    <script src="gamescript.js"></script>
</body>
</html>
```

5.2. CSS Styling

- Uses CSS Grid for game board layout and responsive design.
- Defines fonts (Biryani), colors, gradients, and animations to create a visually appealing interface.

- Implements transitions and hover effects to enhance user interaction.

gameStyle.css

```
@import
url('https://fonts.googleapis.com/css2?family=Biryani:wght@800&display=swap');

* {
   font-family: 'Biryani', sans-serif;
   box-sizing: border-box;
   transition: all 0.3s ease;
}

html, body {
   width: 100vw;
   height: 100vh;
   margin: 0;
   display: flex;
   flex-direction: column;
   align-items: center;
   justify-content: center;
   background: linear-gradient(135deg, #3a1c71 0%, #d76d77 50%, #ffaf7b 100%);
   animation: backgroundGradient 15s ease infinite;
   background-size: 400% 400%;
}

@keyframes backgroundGradient {
   0% { background-position: 0% 50%; }
   50% { background-position: 100% 50%; }
   100% { background-position: 0% 50%; }
}

header {
   background: linear-gradient(135deg, rgba(12, 20, 234, 0.563), rgba(250, 174, 50,
0.563));
```

```
    backdrop-filter: blur(10px);
    padding: 20px;
    border-radius: 10px;
    text-align: center;
    width: 80%;
    margin-bottom: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
}

#logo {
    font-size: 32px;
    color: #fff;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.4);
}

#stats {
    margin-top: 10px;
    display: flex;
    justify-content: space-between;
    width: 100%;
    max-width: 400px;
}

#moves, #time {
    font-size: 18px;
    color: #fff;
}

main {
    width: 80%;
    display: flex;
    justify-content: center;
```

```css
    margin-bottom: 20px;
}

table {
    margin-top: 20px;
    border-spacing: 10px;
}

td {
    background-color: rgba(255, 255, 255, 0.8);
    width: 100px;
    height: 100px;
    perspective: 1000px;
    cursor: pointer;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
    transition: transform 0.6s, box-shadow 0.6s;
}

td:hover {
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.4);
}

.inner {
    position: relative;
    width: 100%;
    height: 100%;
    text-align: center;
    transition: transform 0.8s;
    transform-style: preserve-3d;
}

.front, .back {
    position: absolute;
    width: 100%;
    height: 100%;
    backface-visibility: hidden;
    border-radius: 10px;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 48px;
}

.front {
    background-color: #34ace0;
    transform: rotateY(0deg);
}

.back {
    background-color: #ff793f;
    transform: rotateY(180deg);
}

button {
    background: linear-gradient(135deg, #06a178, #48dbfb);
    border: none;
    border-radius: 10px;
    color: #fff;
    font-size: 18px;
    margin: 10px;
    padding: 10px 20px;
    cursor: pointer;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
    transition: background-color 0.3s, box-shadow 0.3s;
}

button:hover {
    background: linear-gradient(135deg, #48dbfb, #06a178);
    color: black;
    box-shadow: 3px 3px 10px rgba(187, 181, 181, 0.8);
}

#overlay {
    position: absolute;
    top: 0;
    left: 0;
    width: 100vw;
    height: 100vh;
    background-color: rgba(0, 0, 0, 0.7);
    color: white;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 24px;
    z-index: 2;
    padding: 20px;
    box-sizing: border-box;
}

.instructions {
    background-color: antiquewhite;
    color: black;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.5);
    text-align: center;
}

#exitButton {
    background: linear-gradient(135deg, #ff4d4d, #ee0e0e);
    color: white;
    border: none;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: block;
    font-size: 16px;
    margin: 10px auto;
    cursor: pointer;
    border-radius: 12px;
}

#exitButton:hover {
    background: linear-gradient(135deg, #ee0e0e, #ff4d4d);
    color: black;
    box-shadow: 3px 3px 10px rgba(119, 118, 118, 0.8);
}
```

5.3. JavaScript Logic

- Initializes the game on window load, populating tiles with randomized emoji pairs based on selected grid size.
- Handles tile click events to flip tiles, check for matches, update game statistics, and manage game flow.
- Implements timer functionality to track game duration and update the UI accordingly.
- Includes functions for game start, restart, and exit.

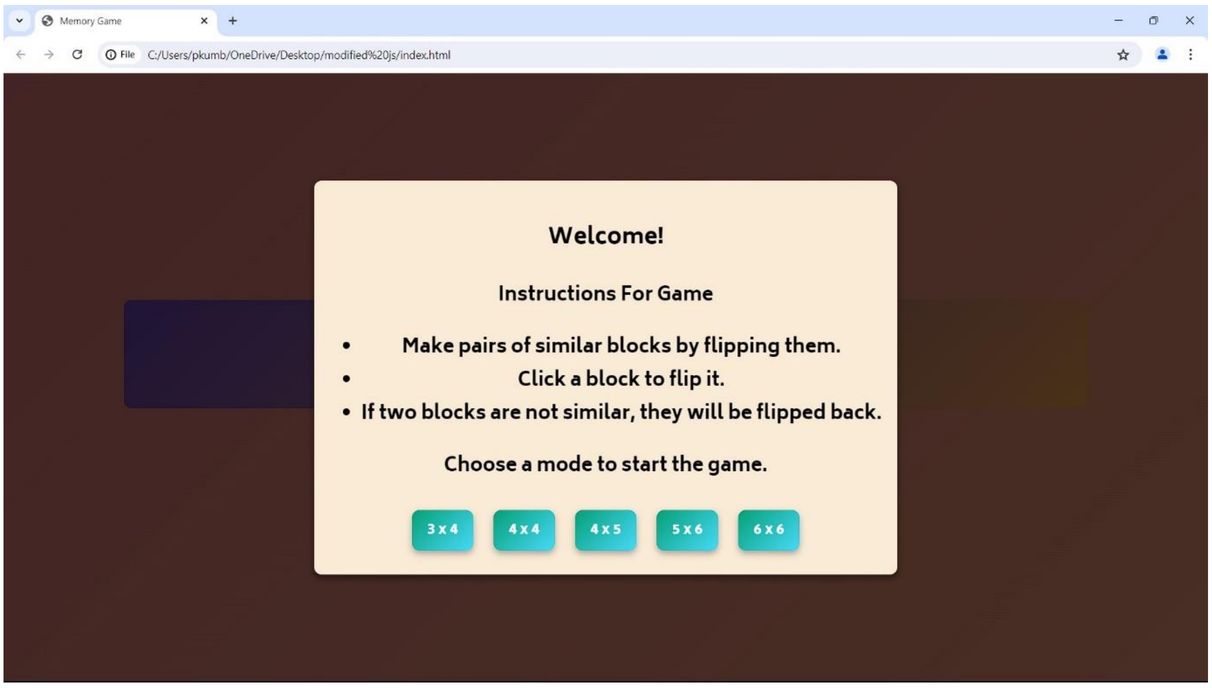
gameScript.js

```
var em = ["💐", "🌹", "🌻", "🌸", "🌺", "🌴", "🌈", "🍓", "🍒", "🍎", "🍉", "🍊", "□",
"🍍", "🍋", "🍏", "🍐", "🥝", "🍇", "□", "🍅", "🌶️", "🍄", "□", "□", "🥑", "🍔", "🍕",
"□", "🎂", "🍬", "🍩", "🍫", "🎈"];

var tmp, c, p = em.length;

if (p) while (--p) {

    c = Math.floor(Math.random() * (p + 1));

    tmp = em[c];
    em[c] = em[p];

    em[p] = tmp;

}

var pre = "", pID, ppID = 0, turn = 0, t = "transform", flip = "rotateY(180deg)", flipBack
= "rotateY(0deg)", time, mode;

window.onresize = init;

function init() {

    W = innerWidth;

    H = innerHeight;

    $('body').height(H + "px");

    $('#overlay').height(H + "px");

}


window.onload = function () {

    console.log("Window loaded");

    $("#overlay").html(`

        <div class="instructions">

            <h3>Welcome!</h3>

            <p>Instructions For Game</p>

            <ul>

                <li>Make pairs of similar blocks by flipping them.</li>

                <li>Click a block to flip it.</li>

                <li>If two blocks are not similar, they will be flipped back.</li>

            </ul>

            <p>Choose a mode to start the game.</p>

            <button onclick="start(3, 4)">3 x 4</button>
            <button onclick="start(4, 4)">4 x 4</button>

            <button onclick="start(4, 5)">4 x 5</button>

            <button onclick="start(5, 6)">5 x 6</button>

            <button onclick="start(6, 6)">6 x 6</button>

        </div>`);

    $("#overlay").show();

}

function start(r, l) {

    min = 0, sec = 0, moves = 0;

    $("#time").html("Time: 00:00");

    $("#moves").html("Moves: 0");

    time = setInterval(function () {

        sec++;

        if (sec == 60) {

            min++; sec = 0;

        }

        if (sec < 10)

            $("#time").html("Time: 0" + min + ":0" + sec);

        else

            $("#time").html("Time: 0" + min + ":" + sec);

    }, 1000);

    rem = r * l / 2, noItems = rem;

    mode = r + "x" + l;

    var items = [];

    for (var i = 0; i < noItems; i++)
```

```
        items.push(em[i]);

    for (var i = 0; i < noItems; i++)

        items.push(em[i]);

    var tmp, c, p = items.length;

    if (p) while (--p) {

        c = Math.floor(Math.random() * (p + 1));

        tmp = items[c];

        items[c] = items[p];

        items[p] = tmp;

    }

    $("table").html("");

    var n = 1;

    for (var i = 1; i <= r; i++) {

        $("table").append("<tr>");

        for (var j = 1; j <= l; j++) {

            $("table").append(`<td id='${n}' onclick="change(${n})"><div
class='inner'><div class='front'></div><div class='back'><p>${items[n -
1]}</p></div></div></td>`);

            n++;

        }

        $("table").append("</tr>");

    }

    $("#overlay").fadeOut(500);

}

function change(x) {

    let i = "#" + x + " .inner";
    let f = "#" + x + " .inner .front";
    let b = "#" + x + " .inner .back";
    if (turn == 2 || $(i).attr("flip") == "block" || ppID == x) { }
    else {
        $(i).css(t, flip);
        if (turn == 1) {
            turn = 2;
            if (pre != $(b).text()) {
                setTimeout(function () {
                    $(pID).css(t, flipBack);
                    $(i).css(t, flipBack);
                    ppID = 0;
                }, 1000);
            }
            else {
                rem--;
                $(i).attr("flip", "block");
                $(pID).attr("flip", "block");
            }
            setTimeout(function () {
                turn = 0;
                moves++;
                $("#moves").html("Moves: " + moves);
            }, 1150);
        }
```

```
        else {
            pre = $(b).text();
            ppID = x;
            pID = "#" + x + " .inner";
            turn = 1;
        }
        if (rem == 0) {
            clearInterval(time);
            if (min == 0) {
                time = `${sec} seconds`;
            }
            else {
                time = `${min} minute(s) and ${sec} second(s)`;
            }
            setTimeout(function () {
                $("#overlay").html(`
                    <div>
                        <h2>Congrats!</h2>
                        <p>You completed the ${mode} mode in ${moves} moves. It took you
${time}.</p>
                        <p>Comment Your Score!<br/>Play Again or Exit?</p>
                        <button onclick="start(3, 4)">3 x 4</button>
                        <button onclick="start(4, 4)">4 x 4</button>
                        <button onclick="start(4, 5)">4 x 5</button>
                        <button onclick="start(5, 6)">5 x 6</button>
                        <button onclick="start(6, 6)">6 x 6</button>
                        <button id="exitButton" onclick="exitGame()">Exit</button>
                    </div>`);
                $("#overlay").fadeIn(750);
            }, 1500);
        }
    }
}

function exitGame() {
    $("#overlay").html(`
        <div style="
            font-size: 56px;
            font-family: 'Trebuchet MS', sans-serif;
            color: #ffcc00;
        ">
            THANK YOU FOR PLAYING!
        </div>
    `);
    $("#overlay").fadeIn(750);  // Ensure the overlay is visible
}
```

OUTPUTS



Memory Game
File C:/Users/pkumb/OneDrive/Desktop/modified%20js/index.html

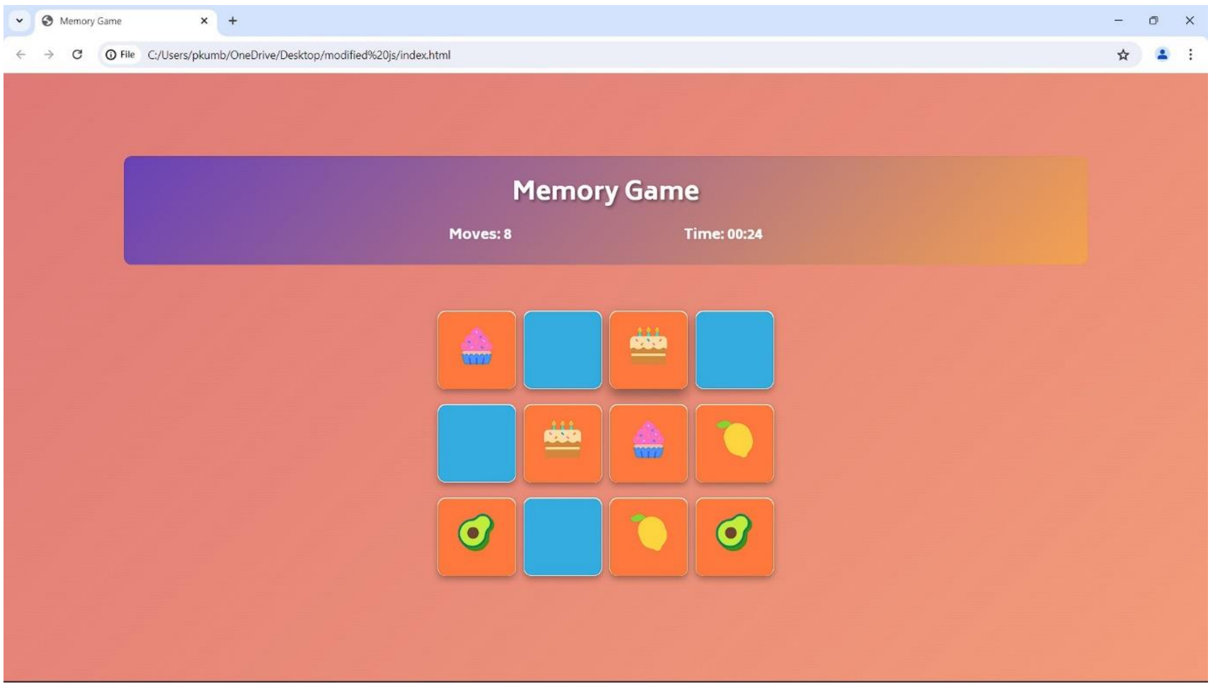
Welcome!

Instructions For Game

- Make pairs of similar blocks by flipping them.
- Click a block to flip it.
- If two blocks are not similar, they will be flipped back.

Choose a mode to start the game.

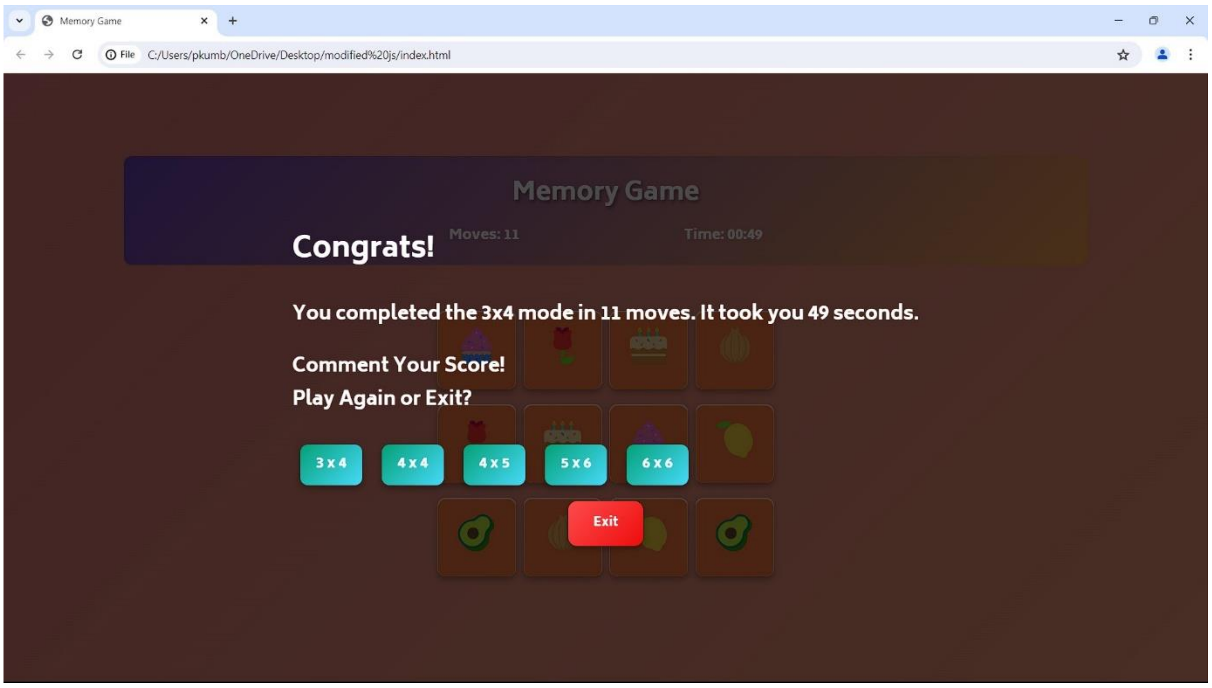
3 x 4
4 x 4
4 x 5
5 x 6
6 x 6



Memory Game
File C:/Users/pkumb/OneDrive/Desktop/modified%20js/index.html

Memory Game

Moves: 8 Time: 00:24

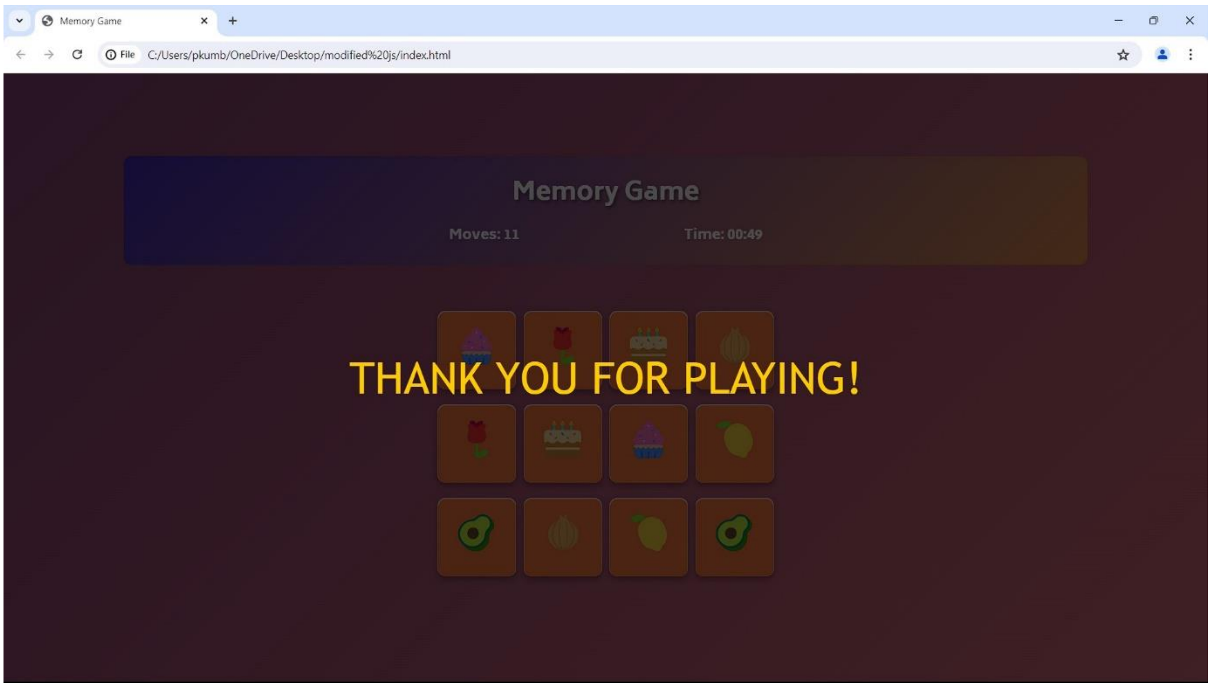


Congrats!

You completed the 3x4 mode in 11 moves. It took you 49 seconds.

Comment Your Score!

Play Again or Exit?



THANK YOU FOR PLAYING!

6. Challenges Faced

- **Implementation of Game Logic:** Ensuring tiles flip correctly, match detection, and managing game state transitions.
- **Responsive Design:** Achieving consistent layout and usability across various devices and screen sizes.

- **Performance Optimization:** Ensuring smooth animations and interactions without compromising browser performance.

7. Future Enhancements

- **Scoreboard:** Implement a scoreboard to track and display high scores.
- **Difficulty Levels:** Introduce different difficulty levels with varying grid sizes and tile counts.
- **Accessibility:** Improve accessibility features such as keyboard navigation and screen reader compatibility.

8. Conclusion

The Memory Matching Game project successfully creates an interactive and enjoyable gaming experience through the integration of HTML, CSS, and JavaScript technologies. It achieves the project objectives by providing a visually appealing interface, engaging gameplay mechanics, and responsive design. Future enhancements can further enrich the game's features and accessibility.

9. References

- Google Fonts: https://fonts.google.com/
- jQuery Documentation: https://api.jquery.com/