Memory Match Challenge

An Interactive Web-Based Card Matching Game

Course: Web Programming
Student: Samar Mohammed Balousha

Academic Number:2320222601
Date: August 2025
Instructor: Maamoun Abu Salah

Live Demo:

https://samarbal.github.io/Memory-Game/

GitHub Repo:

https://github.com/Samarbal/Memory-Game

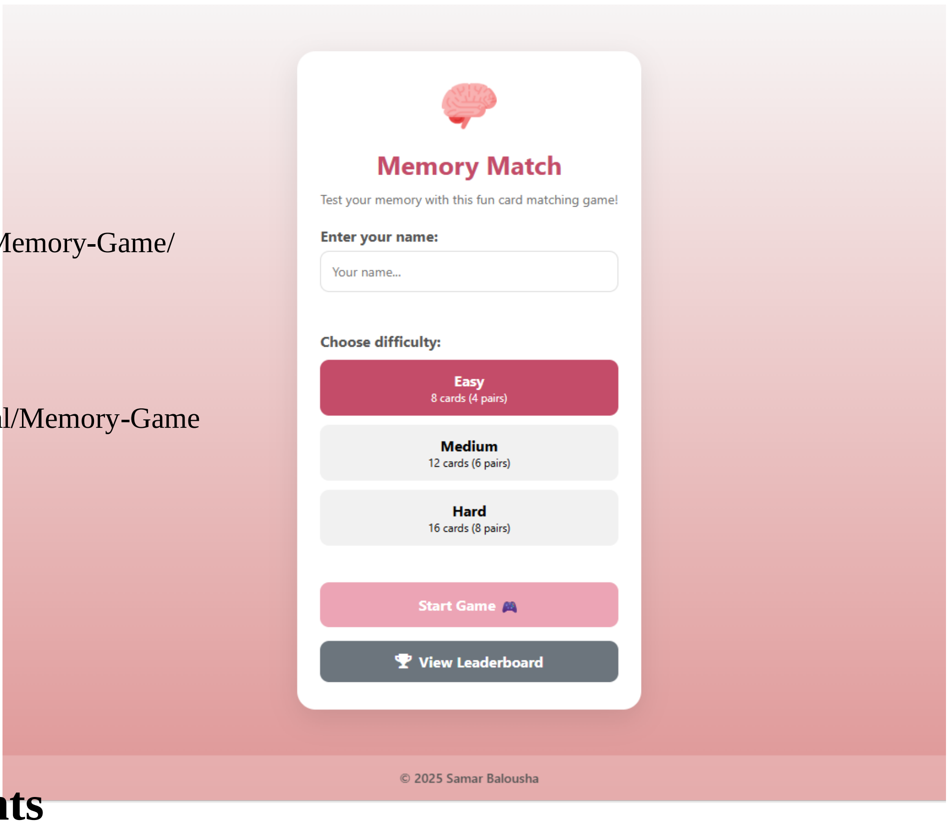


Table of Contents

1. Project Overview

1.1 Project Title

Memory Match Challenge — Interactive Web-Based Card Matching Game

1.2 Project Description

The **Memory Match Challenge** is a browser-based card-matching game that tests players' memory through simple but engaging gameplay. It has two main parts: a **Main Menu** where users enter their name, pick a difficulty (Easy, Medium, Hard), and view the leaderboard, and a **Game Page** where the matching takes place.

Cards start face-down and are flipped two at a time; matches stay revealed, while mismatches flip back after a short delay. The goal is to find all pairs in the fewest moves and shortest time.

The game features a **responsive design** for laptop and mobile, smooth **3D card animations**, and a **win screen** showing stats. Progress (name, difficulty, time, and moves) is stored in **localStorage**, powering a leaderboard that ranks top players.

Built with **HTML, CSS, and JavaScript**, it demonstrates practical skills in DOM manipulation, animations, event handling, and client-side data storage.

1.3 Problem Statement

Many online memory games are either overly complex, cluttered with ads, or not optimized for smooth play on all devices. Players looking for a quick, fun, and distraction-free way to train their memory often end up frustrated by poor design, laggy animations, or lack of simple progress tracking.

1.4 Solution Approach

I created a clean, and responsive memory matching game that works seamlessly on both laptop and mobile. It focuses on smooth gameplay, simple controls, and visual feedback that keeps players engaged. With built-in difficulty levels, a real-time timer, and a local leaderboard, players can challenge themselves and track their improvement.

1.5 Target Audience

The game is for anyone who enjoys quick, fun, and brain-challenging activities — from kids testing their memory to adults chasing top leaderboard spots — playable on any device with a browser.

2. Technical Specifications

2.1 Tech Stack

A lightweight pure HTML, CSS, and JavaScript stack used for this project, ensuring fast load times and simple deployment.

2.2 Frontend Technologies

- **Framework/Library:** None — implemented using Vanilla JavaScript for full control over the DOM and game logic.
- **Styling:** Pure CSS3 using **Flexbox** and **CSS Grid** for responsive layouts.
- **Additional Tools:**
  - **Google Fonts** for typography.
  - **Font Awesome** for UI icons.
  - **CSS animations** for smooth 3D card flip effects and modal transitions.

2.3 Backend Technologies

- **Programming Language:** JavaScript (for both game logic and data handling).
- **Database:** No external database; **localStorage** is used to store and retrieve player data (username, difficulty, score history, leaderboard).

2.4 Development Tools & Environment

- **Version Control:** Git + GitHub
- **Code Editor:** Visual Studio Code
- **AI-Powered Development Tools:**
  - **Lovable** for initial UI layout design.
  - **Cursor AI** for code refactoring, optimization, and debugging assistance.

3. Project Features & Functionality

3.1 Core Features

1. **Main Menu with Player Setup** – Players can enter their username, choose a difficulty (Easy, Medium, Hard), and access the leaderboard.
2. **Dynamic Card Generation** – Game board layout changes based on difficulty, with cards randomly shuffled each game.
3. **Interactive Card Flipping** – Smooth 3D flip animations for revealing and hiding cards.
4. **Scoring System** – Tracks time and moves taken, with results saved per player and difficulty level.
5. **Local Leaderboard** – Displays the top players' scores stored in localStorage, ranked by time and moves.

3.2 Advanced Features

- **Win Screen Modal** – Shows detailed results after game completion (time, moves, difficulty) with quick actions to replay or view leaderboard.
- **Responsive Design** – Works across laptops and mobile devices, adjusting card sizes and grid layout dynamically.
- **Error Handling** – Detects missing DOM elements and invalid user inputs to prevent game crashes.

4. AI Tools Integration

From the very start of this project, AI tools played a big role in shaping both the design and the code.

I first used **Lovable** to quickly create a clean, attractive game UI. This gave me a visual starting point and saved hours of manual design work. Once I had that base, I implemented the **HTML** and **CSS** myself, making sure every element matched the intended style and was fully responsive.

When it came to **JavaScript**, I began by writing the core game logic — setting up variables, functions, conditions, and loops. At this stage, I turned to **Cursor AI** as a coding partner. It helped me catch errors early, suggested more efficient ways to write certain functions, and even helped me make my code cleaner and easier to read.

In short, AI tools didn't replace my work — they **boosted my productivity, reduced debugging time**, and allowed me to focus on making the game experience smooth and fun for players.

5. Expected Challenges & Solutions

5.1 Challenges

1. **Making the game responsive across all devices** – Ensuring the layout and animations work well on mobile, tablets, and laptops.
2. **Keeping game state bug-free** – Managing flipped cards, matches, timers, and move counters without unexpected glitches.
3. **Sorting and storing scores efficiently** – Making sure the leaderboard shows accurate, sorted results and keeps only the best ones.
4. **Hosting the game online** – Making the game accessible online for testing and presentation

---

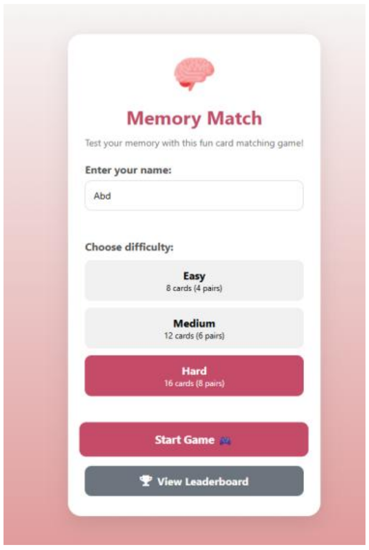
5.2 Proposed Solutions

- **Responsive Design:** Use a **mobile-first approach**, CSS Grid/Flexbox, and media queries so that the game naturally adapts to different screen sizes.
- **Game State Management:** Keep all critical game values (like flipped cards, matches, moves) in a **centralized game state object** to avoid confusion and make debugging easier.
- **Leaderboard Logic:** Use **localStorage** with sorting by time first and moves second, while limiting the saved results to the top 50 to prevent clutter.
- I deployed the project using **GitHub Pages**, which provided a free and simple hosting solution: https://samarbal.github.io/Memory-Game/.

6. System Workflow

Below is exactly how the app runs—from the first page load to saving a winning score—written in the same sequence your code executes.

Step 1: Initial Page Load (Main Menu)


Memory Match
Test your memory with this fun card matching game!
Enter your name:
Abd
Choose difficulty:
Easy
8 cards (4 pairs)
Medium
12 cards (6 pairs)
Hard
16 cards (8 pairs)

- **User opens** index.html**.**
- The browser loads HTML, CSS, and JavaScript files.
- The main menu is displayed:
  - Game title, icon, and subtitle.
  - Username input field.
  - Difficulty selection (Easy, Medium, Hard).
  - Start Game and Leaderboard buttons.

Step 2: User Setup

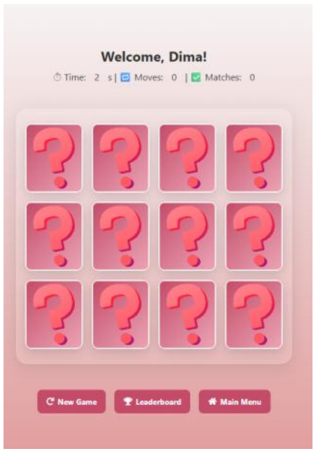
- **User enters their name** in the input field.
- **User selects a difficulty** by clicking one of the options (active state toggles).

- **User clicks "Start Game":**
    - JavaScript validates the username (not empty, ≤15 characters).
    - If invalid, a warning message is shown.
    - If valid:
        - Username and selected difficulty are saved to localStorage.
        - Browser navigates to gamePage1.html.

Step 3: Game Page Initialization

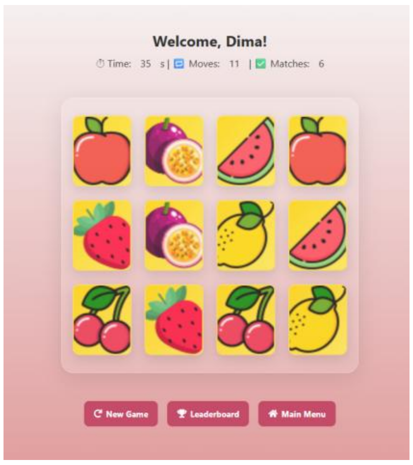
- gamePage1.html **loads.**
- JavaScript detects it's the game page and runs initializeGame().
- **Username and difficulty** are retrieved from localStorage.
- **Header displays:** Welcome message, timer (0s), moves (0), matches (0).
- **Game board is generated:**
    - Number of pairs depends on difficulty (4, 6, or 8).
    - Card images are selected, duplicated, shuffled, and rendered as clickable cards (face-down).
- **Control buttons** ("New Game", "Leaderboard", "Main Menu") are set up.

Step 4: Game Start



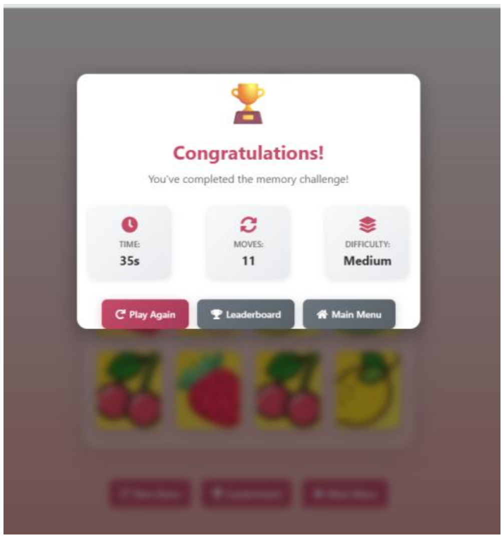
- **Timer starts** counting seconds.
- **User clicks cards** to flip them:
    - First card flips to show its image.
    - Second card flips.
    - If images match:
        - Both cards stay face-up, marked as matched.
        - Matches counter increases.
    - If not:
        - Both cards flip back after a short delay.
    - Moves counter increases after every pair flipped.

Step 5: Game Progress



- **User continues flipping cards** to find all pairs.
- **Game state** (flipped cards, matches, moves, timer) is tracked in memory.
- **Display updates** in real-time (timer, moves, matches).

Step 6: Winning the Game



Congratulations!

You've completed the memory challenge!

TIME: 35s

MOVES: 11

DIFFICULTY: Medium

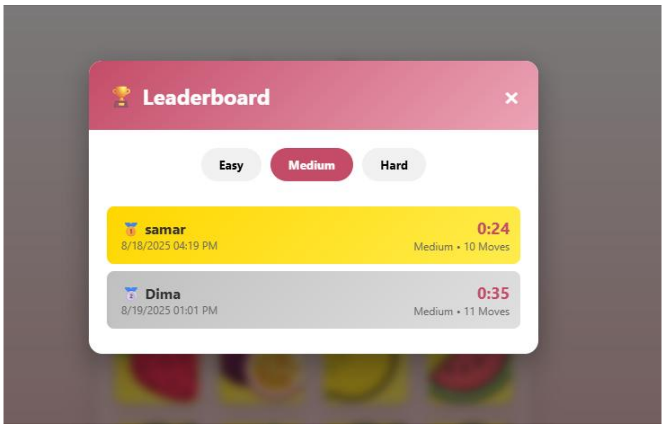
- **When all pairs are matched:**
  - Timer stops.
  - Game state is locked (no more moves).
  - **Score is saved** to localStorage:
    - Username, difficulty, time, moves, date.
    - Scores are sorted by time and moves; only top 50 are kept.
  - **Win screen modal appears:**
    - Shows stats (time, moves, difficulty).
    - Offers buttons: Play Again, Leaderboard, Main Menu.

Step 7: Post-Game Actions

- **User can:**
  - **Play Again:** Restarts the game (reloads page).
  - **View Leaderboard:** Opens leaderboard modal, showing top scores for selected difficulty.
  - **Return to Main Menu:** Navigates back to index.html.

Step 8: Leaderboard Modal

- **Leaderboard modal** can be opened from either page.
- **Difficulty tabs** let user switch between Easy, Medium, Hard leaderboards.
- **Scores are displayed** with medals for top 3, showing player name, time, moves, and date.



samar 8/18/2025 04:19 PM 0:24 Medium • 10 Moves

Dima 8/19/2025 01:01 PM 0:35 Medium • 11 Moves

Step 9: Responsive & Error Handling

- **UI adapts** to laptops, tablet, and mobile via CSS.
- **JavaScript checks** for missing DOM elements and invalid inputs, preventing crashes.

7. File-by-File Code Explanation

1. HTML Files

index.html (Main Menu Page)

- Sets up the main menu where users enter their name, choose a difficulty, and start the game.
- Includes buttons for starting the game and viewing the leaderboard.
- Contains a hidden game board (not used here), and a modal for the leaderboard.
- Footer credits the author.

gamePage1.html (Game Page)

- Displays the game interface after the user starts a game.
- Shows the player's name, timer, moves, and matches.
- Contains the game board where cards are displayed and flipped.
- Includes controls for new game, leaderboard, and returning to main menu.
- Modals for leaderboard and win screen pop up as needed.
- Footer credits the author.

2. CSS (gameStyle.css)

- Uses Flexbox and CSS Grid to lay out the main menu and game board responsively.
- Styles buttons, cards, and modals with colors, rounded corners, and hover effects.
- Animates card flips with 3D transforms, and modals with slide-in and bounce effects.
- Adapts layout and sizes for tablets and mobile devices using media queries.

3. JavaScript (gameScript.js)

initializeMainMenu

- Sets up difficulty selection and validates username input.
- Saves username and difficulty to localStorage and starts the game.
- Handles leaderboard modal display on the main menu.

initializeGame

- Loads username and difficulty from localStorage.
- Sets up the game board with shuffled cards based on difficulty.
- Initializes timer, moves, matches, and control buttons.

createCard

- Generates a card element with front and back faces for flipping animation.

flipCard

- Adds or removes the "flipped" class to show/hide the card's front image.

handleCardClick

- Manages card flipping, match checking, and move counting.
- Locks flipping during animations and prevents flipping matched/disabled cards.

markCardAsMatched / disableCard

- Marks cards as matched and disables further interaction.

handleGameWin

- Stops the timer, saves the score, and shows the win screen modal.

initializeBoard

- Selects images, shuffles them, and populates the game board with card elements.

initializeTimer

- Starts and updates the game timer every second.

initializeControls

- Sets up event listeners for new game and main menu buttons.

saveScore

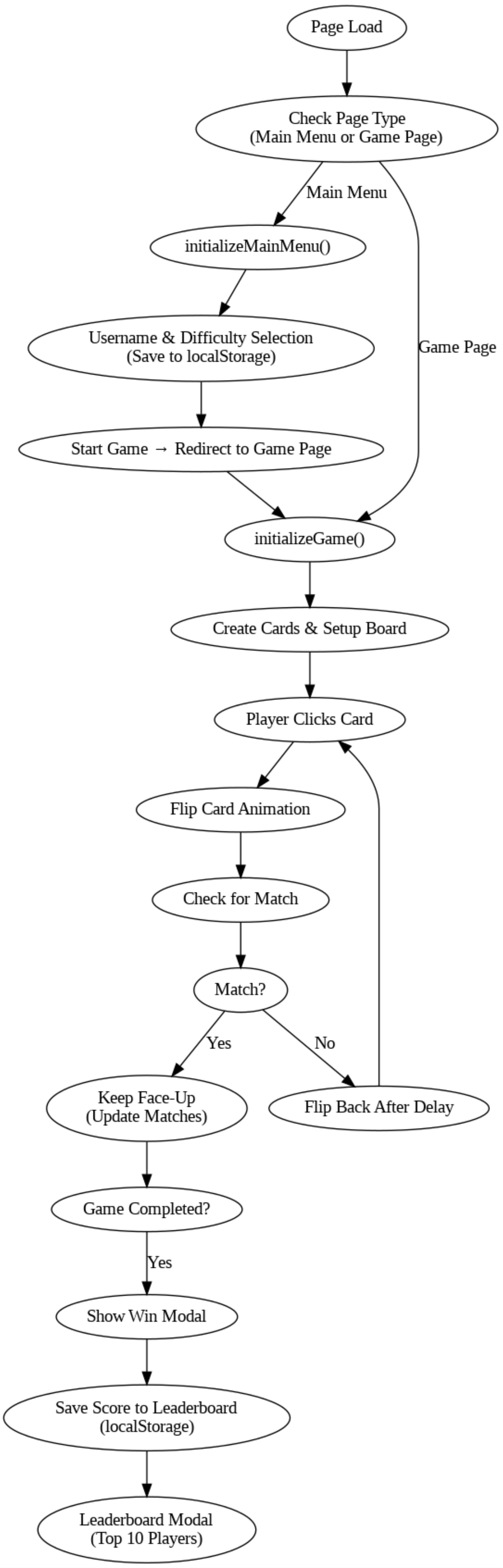
- Saves the player's score (name, difficulty, time, moves) to localStorage, keeping only the top 50.

getTopScores

- Retrieves and sorts scores for the selected difficulty from localStorage.

displayLeaderboard

- Shows the top scores in the leaderboard modal, with medals for top ranks.


Page Load
Check Page Type (Main Menu or Game Page)
Main Menu
Game Page
initializeMainMenu()
Username & Difficulty Selection (Save to localStorage)
Start Game → Redirect to Game Page
initializeGame()
Create Cards & Setup Board
Player Clicks Card
Flip Card Animation
Check for Match
Match?
Yes
No
Keep Face-Up (Update Matches)
Flip Back After Delay
Game Completed?
Yes
Show Win Modal
Save Score to Leaderboard (localStorage)
Leaderboard Modal (Top 10 Players)

showLeaderboard / hideLeaderboard

- Opens or closes the leaderboard modal.

initializeLeaderboard

- Sets up tab switching and modal closing for the leaderboard.

showWinScreen / hideWinScreen / initializeWinScreen

- Displays the win screen modal with stats and actions (play again, leaderboard, main menu).

8. Testing & Results

The game was tested on multiple devices (laptop, and mobile) to ensure responsive design and smooth performance. All features — including card flipping, scoring, timer, and leaderboard — worked as expected without major bugs. The deployment on GitHub Pages allowed additional testing online, confirming that the game runs smoothly across browsers like Chrome and Firefox.

9. Conclusion

The *Memory Match Challenge* demonstrates my ability to build a fully functional, responsive web game using HTML, CSS, and JavaScript. Through this project, I strengthened my problem-solving and debugging skills while learning to manage game state and user interaction effectively. It also gave me practical experience with deployment and AI-assisted development.