## ****Title Page****

**Memory Match Challenge**  
An Interactive Web-Based Card Matching Game

**Course:** Web Programming  
**Student:** Samar Balousha  
**Date:** August 2025  
**Instructor:** [Instructor Name]

Insert a screenshot of the first page

## ****Table of Contents****

1. **Project Overview**  
    1.1 Project Title  
    1.2 Project Description  
    1.3 Problem Statement  
    1.4 Solution Approach  
    1.5 Target Audience
2. **Technical Specifications**
3. **Project Features & Functionality**
4. **AI Tools Integration**
5. **Expected Challenges & Solutions**
6. **System Workflow**
7. **File-by-File Code Explanation**
8. **Testing & Results**
9. **Conclusion & Reflection**

## ****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 fun, browser-based game that puts your memory skills to the test through a simple but addictive card-matching challenge.

The game is split into two main parts:

* **Main Menu** – where you type in your name, pick a difficulty (Easy, Medium, Hard), and check out the leaderboard.
* **Game Page** – where the real action happens.

When you start, all cards are face-down in a grid that changes size depending on the difficulty you picked. On each turn, you flip two cards. If they match, they stay face-up; if not, they flip back after a short pause. Your mission is to find all the pairs as quickly as possible and in the fewest moves.

It’s built with a responsive design so it works smoothly on both desktop and mobile. The cards flip with smooth 3D animations, and when you win, you get a pop-up screen showing your stats. Your progress — including username, difficulty, time, and moves — is saved using localStorage, and the leaderboard ranks the top players for each difficulty.

Behind the scenes, this project uses HTML, CSS, and JavaScript to handle animations, DOM interactions, event handling, data storage, and game state management. It’s not just fun to play — it’s also a showcase of practical web development skills.

### **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 desktop 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 was used for this project, ensuring fast load times and simple deployment without the complexity of frameworks.

### **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 ([Repository Link](https://github.com/Samarbal/Memory-Game))
* **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 desktop 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 Potential Challenges**

1. **Making the game responsive across all devices** – Ensuring the layout and animations work well on mobile, tablets, and desktops.
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.

### **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.

# 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.

Memory Game Project

**Project Overview**

**Project Title**

Memory Match Challenge — Interactive Web-Based Card Matching Game

**Project Description**

The **Memory Match Challenge** is a fun, browser-based game that puts your memory skills to the test through a simple but addictive card-matching challenge.

The game is split into two main parts:

* **Main Menu** – where you type in your name, pick a difficulty (Easy, Medium, Hard), and check out the leaderboard.
* **Game Page** – where the real action happens.

When you start, all cards are face-down in a grid that changes size depending on the difficulty you picked. On each turn, you flip two cards. If they match, they stay face-up; if not, they flip back after a short pause. Your mission is to find all the pairs as quickly as possible and in the fewest moves.

It’s built with a **responsive design** so it works smoothly on both desktop and mobile. The cards flip with **smooth 3D animations**, and when you win, you get a pop-up screen showing your stats. Your progress — including username, difficulty, time, and moves — is saved using **localStorage**, and the **leaderboard** ranks the top players for each difficulty.

Behind the scenes, this project uses **HTML, CSS, and JavaScript** to handle animations, DOM interactions, event handling, data storage, and game state management. It’s not just fun to play — it’s also a showcase of practical web development skills.

**Technical Specifications**

**Tech Stack**

A lightweight **pure HTML, CSS, and JavaScript** stack was used for this project, ensuring fast load times and simple deployment without the complexity of frameworks.

**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** icons for UI elements.
  + **CSS animations** for smooth 3D card flip effects and modal transitions.

**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).

**Development Tools & Environment**

 **Version Control:** Git + GitHub (<https://github.com/Samarbal/Memory-Game> )

 **Code Editor:** Visual Studio Code.

 **AI-Powered Development Tools:** Cursor AI for code suggestions, refactoring, and debugging assistance.

 **Package Manager:** None — no external dependencies were required for this version.

**Project Features & Functionality**

**Core Features**

 **Main Menu with Player Setup** – Players can enter their username, choose a difficulty (Easy, Medium, Hard), and access the leaderboard.

 **Dynamic Card Generation** – Game board layout changes based on the chosen difficulty, with cards randomly shuffled each game.

 **Interactive Card Flipping** – Smooth 3D flip animations for revealing and hiding cards.

 **Scoring System** – Tracks time and moves taken, with results saved per player and difficulty level.

 **Local Leaderboard** – Displays the top players’ scores stored in localStorage, ranked by time and moves.

**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 desktop and mobile devices, adjusting card sizes and grid layout dynamically.

 **Error Handling** – Checks for missing DOM elements and invalid user input to prevent crashes

**AI Tools Integration**

To speed up development and enhance code quality, I incorporated AI-powered tools throughout the project:

* **Lovable** was used in the early stage to design and generate the **initial game UI layout**, providing a ready-to-use visual framework for the memory game interface.
* I then manually implemented the HTML/CSS structure based on this design.
* During JavaScript development, I used **Cursor AI** to refine the **final look and styling**, ensuring the design remained responsive and visually appealing.
* While building the game logic (functions, variables, conditions, and loops), I leveraged **Cursor AI** to **optimize, debug, and improve** my code for better readability, maintainability, and performance.

By combining my own development work with AI-assisted enhancements, I was able to create a cleaner, more efficient, and polished final product.

**Expected Challenges & Solutions**

**Potential Challenges**

 **Responsive Design Across All Devices** – Ensuring that the game layout, animations, and modals look good on mobile, tablets, and desktop screens.

 **Efficient Game State Management** – Keeping track of flipped cards, matches, timer, and moves without bugs or unexpected behavior.

**Proposed Solutions**

 **Responsive Design** – Use CSS flex/grid layouts, media queries, and mobile-first design to adapt elements dynamically to different screen sizes.

 **Game State Management** – Use clear game state variables, modular functions, and error handling to prevent mismatches or timer issues.

**System Workflow**

The **Memory Match Challenge** follows a straightforward yet interactive flow:

1. **Main Menu**
   * When the player visits the site, they are greeted with a main menu.
   * The player **enters their username**, selects a difficulty level (Easy, Medium, Hard), and can optionally view the leaderboard before starting.
   * Input validation ensures the username is not empty and within length limits.
2. **Game Initialization**
   * Upon clicking “Start Game,” the selected **username** and **difficulty** are stored in localStorage.
   * The game page is loaded, and the board is generated dynamically based on difficulty:
     + **Easy** → 4 pairs (8 cards)
     + **Medium** → 6 pairs (12 cards)
     + **Hard** → 8 pairs (16 cards)
   * Cards are **shuffled randomly** before being displayed face-down.
3. **Gameplay**
   * The player clicks on cards to flip them and try to match pairs.
   * If two flipped cards match, they remain face-up. If not, they flip back after a short delay.
   * The system **tracks time and move count** during the game.
4. **Win Condition**
   * When all pairs are matched, the game stops the timer.
   * A **Win Screen Modal** appears, showing:
     + Total time taken
     + Number of moves
     + Chosen difficulty
   * The player can choose to replay, return to the main menu, or view the leaderboard.
5. **Score Saving & Leaderboard**
   * The player’s score (username, difficulty, time, moves, and date) is saved in localStorage.
   * Scores are **sorted by best time**, then by fewer moves.
   * The leaderboard displays the **top 10 players** for each difficulty level.
6. **Data Persistence**
   * All saved scores remain even if the browser is refreshed or closed, thanks to localStorage.
   * This ensures players can track their progress over time.

**Resources & References**

**Documentation & Tutorials**

[List any specific documentation, tutorials, or learning resources you plan to use]

**Inspiration/Similar Projects**

[Mention any existing projects or websites that inspired your idea]

**Additional Notes**

[Any additional information, special requirements, or considerations for your project]