# Pokémon Card Game Demo Link

https://velvety-mousse-38f1cf.netlify.app/

# Objectives

The objective of this assignment is to design and develop a memory card game using CSS and JS. Instructions:

Create a card game utilizing Pokémon cards.

Retrieve Pokémon randomly from the Pokiapi.co API server.

Introduce game difficulty levels: easy, medium, and hard. Adjust the number of cards and time limits according to the chosen difficulty. Failure to complete the game within the specified time will result in a loss for the player.

Ensure that each Pokémon can only be assigned to a single card and cannot be used for multiple cards. Add a start and a reset buttons to the game.

Display a header showing the number of clicks the user has mad, the number of pairs left, number of pairs matched, and total number of pairs.

Allow users to select a themes. For example, a user can choose between dark and light themes. Add a power up feature that allows the player see all the cards for short time

# User Stories

As a player, I want to play a card game utilizing Pokémon cards, So that I can enjoy the game with familiar and exciting Pokémon characters.

As a game developer, I want to retrieve Pokémon randomly from the Pokiapi.co API server, So that each game session offers a unique and diverse set of Pokémon cards.

As a player, I want to have different difficulty levels (easy, medium, and hard), So that I can choose the level of challenge that suits my skill and preference.

As a player, I want the number of cards and time limits to adjust based on the chosen difficulty level, So that the game remains balanced and appropriately challenging.

As a player, I want to ensure that each Pokémon can only be assigned to a single card, So that the game follows the rules of memory card games and offers a fair gameplay experience.

As a player, I want the game to have a start button, So that I can initiate a new game session whenever I'm ready.

As a player, I want the game to have a reset button, So that I can restart the current game session if I make a mistake or want to start over.

As a player, I want to see a header displaying relevant game information, Such as the number of clicks I have made, the number of pairs left, the number of pairs matched, and the total number of pairs, So that I can track my progress and stay informed during the game.

As a player, I want to be able to select different themes (e.g., dark and light), So that I can customize the visual appearance of the game according to my preference.

As a player, I want a power-up feature that allows me to see all the cards for a short period of time, So that I can strategically plan my moves and improve my chances of finding matching pairs.

# Suggested Development Strategy

Step 1. Setup the files and the imports

Step 2. Create Six cards. Add two images inside each card; front and back in each card. Have the cards inside a flex box.

Step 3. Overlap the front and the back of each card so that only the back of the cards are visible. Step 4. Add a flip animation to the cards on hover.

Step 4.1 Add the flip class to rotateY(180deg) to the .card:hover and transition in 1s Step 4.2 Make the rotation in 3d by adding perspective: 1000px; to the .card

Step 5. Check if two cards are the same.

Step 6a. If two cards are not the same, flip them back

Step 6b. If two cards are not the same, flip them back with some delay Step 7. If two cards are the same, remove them from the game.

Step 8. Corner case 1. If the user clicks on the same card twice, do nothing.

Step 9. Corner case 2. If the user clicks on a card that is already matched, do nothing.

Step 10. Corner case 3. If the user clicks on a card while two cards are already flipped, do nothing. Step 11a. Add the winning event. If user clicks on all the cards, display a winning message.

Step 11b. Add a header showing the number of clicks the user has made, and the number of pairs left, number of pairs matched, and total number of pairs.

Step 12a. Add a timer to the game. Step 12b. Show the time in the header.

Step 13. Add a *reset* button to the game. Step 14. Add a *start* button to the game.

Step 15.a Add a difficulty level to the game. Show the levels controls to the header. Step 15.b Add the logic to the difficulty levels.

Step 16. Add themes

Step 17. Add power-up logic. May be reveal all cards for short period of time.

# Rubric [Out of 50]:

[2 Marks ] [complete/incomplete] Cards flip when clicked

[2 Marks ] [complete/incomplete] If the user clicks on the same card twice, do nothing.

[2 Marks ] [complete/incomplete] If the user clicks on a card that is already matched, do nothing. [2 Marks ] [complete/incomplete] If the user clicks on a card while two cards are already flipped, do nothing.

[7 Marks ] [complete/incomplete] Winning Event is implemented. If user clicks on all the cards, display a winning message.

[7 Marks ] [complete/incomplete] Header is showing the number of clicks the user has made, and the number of pairs left, number of pairs matched, total number of pairs, and game timer.

[7 Marks ] [complete/incomplete] Start and Reset buttons are added to the game. [7 Marks ] [complete/incomplete] Difficulty levels are added to the game.

[7 Marks ] [complete/incomplete] Themes are added to the game.

[7 Marks ] [complete/incomplete] Power-up logic is added to the game.

Deductions

|  |  |
| --- | --- |
| **Requirement** | **Deduction** |
| Missing Hosting Link | -10 [-20%] |
| Missing or Incomplete Checklist | -10 [-20%] |
| Missing GitHub Link | -15 [-30%] |
| Video is 10s to 60s too long | -5 [-10%] |
| Video is >60s too long | -10 [-20%] |
| Missing Video Link | -25 [-50%] |
| Missing atomic and incremental commits | -10 [-20%] |