# Homework 2 Set! Game

# Part 1: Game Appearances.

Millstone NO. 1 Game interface. (HTML/CSS)

Due: 12:59pm, Monday, March 8th.

Total points: 30pts.

We will check your progress on March 5th.

Set! Is my favorite board game. You can have a sense of it by playing this online-version:

<https://www.setgame.com/set/puzzle>

Here is more information about the game.

<https://en.wikipedia.org/wiki/Set_(card_game)>

The basic rule of set is:

You take 3 cards out of 12 cards to form a “set”. A “set” consists of three cards satisfying *all* of these conditions:

* They all have the same number or have three different numbers.
* They all have the same shape or have three different shapes.
* They all have the same shading or have three different shadings.
* They all have the same color or have three different colors.

Once the set is found, these cards are taken away from the board. You have to find the “set” from the rest of the deck. If you can’t find them anymore, you can add more cards (refreshing the board).

In reality, SET can be played with multiple users competitively. But on the web-version, usually one user plays with oneself.

Our goal is make a similar SET game using HTML/CSS/JavaScript.

This homework needs TWO-Parts submission.

**Please do not copy code from INTERNET. If caught, will be reported to academic integrity.**

**In this submission (due March 8th), you are expected to**

**1) Using CSS to code up the Main Menu View.**

**2) Using CSS to code up to the Game View.**

**3) Let user select a timing option and level of difficulty.**

**4) The user can select the minuets and the level of difficulty.**

**5) When the user press “Starts”, it goes to the Game View.**

**6) Feel free to finish as much as additional behavioral of the game as best as you can. But they are not required.**

**7) You are required to present your partial results in class break-out rooms on March 5th.**

We will complete the behavior and the whole game in the next assignment.

Feel free to discuss this project with me, TA, and your classmates. But you must turn in your independent work.

Deliverables:

Please submit your .html, .css, .js and your image folders as a .zip file.

Please name your folder as your firstname\_lastname\_homework2

Diagram, shape

Description automatically generated

This assignment continues what we've been learning in Module 1 (HTML/CSS) and applies what you are learning in Module 2 (JavaScript, DOM, and Animations).

As a longer assignment, it will be worth more points than HW2 and consists of two submissions. First, we will focus on CSS/HMTL for the user views. Second, we will focus on JavaScript to shuffle the cards, let user select the cards and record the scores.

Feel free to complete as much as you can already in the first submission.

Main Menu Appearance (You are given the set.html) you are asked to code up CSS. Feel free to make small changes in HTML as you need.

Graphical user interface, text, application, Teams

Description automatically generated

* The purple box above represents the first article in the #menu-view section - the background color is #6F77ED and it has a 5pt solid black border with a border radius of 0.35em.
* Other than the dropdown and Start button, all text in the purple box is white.
* All elements are horizontally-centered in the Menu View as shown above.
* All h2 elements have underlined text with a font size of 16pt, but all other text in the Menu View has a font size of 12pt.
* The Start button and dropdown have white backgrounds with a 2pt solid black border.
* The Start button has a fixed width of 100px, 10px of margin on the top, and 8px of padding between the inner text and border.
* There is 20px of spacing between the bottom border of the Start button and the bottom border of the purple box.
* No other margin or padding should be overridden from the defaults.
* **Note**: the default options for both the timing dropdown and the difficulty radio buttons are set in the provided HTML document.
* **Note**: The rendering of the select element is particularly different between browser/operating systems. Refer to the expected screenshots on your system, but if you are missing the "up/down" arrows don't worry about this.
* **Game View**
* *Note: After you implement the CSS for the .hidden class correctly, the #game-view should be hidden. But it will be easier to work on the CSS for the Game View before using JS to switch views. To do so, we recommend temporarily hiding the #menu-view in your CSS, and since the HTML comes with the .hidden class added to #game-view, commenting out the .hidden ruleset in your CSS so that the #menu-view is hidden and #game-view is shown. Eventually, you’ll want to toggle these views using using JS as detailed later.*

Diagram

Description automatically generated

##### Details Bar

* When the Game View is visible, there should be a black "details" bar with a board of cards underneath.
* The details bar includes a "Back to Main" button, a set counter, a timer, and a "Refresh Board" button. All four elements should be centered vertically in this bar with their horizontal distribution set to space-evenly.
* The top corners of the bar should be rounded with a border radius of 0.5em.
* Aside from the buttons which have black font, text in the details bar should be white.
* As with the Start button, both buttons should have a white background with 8px of padding between the text and their borders (which should be the same as the Start button). These buttons should also have 12pt font.
* Any other text in this bar has a font size of 14pt.

##### Board

* You should use flex positioning where appropriate to layout the board and cards (do not use float or absolute positioning in this assignment).
* The #game has a black border on the left, right, and bottom with a 5pt width.
* The cards of the #game element should be horizontally distributed using space-evenly.
* There should be 20px of padding on all sides of the #game board (this spacing will be fixed on the top/bottom but there may be extra spacing visible on the left/right depending on how cards fit for a particular screen width).
* Each card should have 5px of top and bottom margin.
* The cards should wrap naturally in the board as screen sizes adjust (hint: with appropriate flex properties, you don’t need to define any rows or columns).
* Whereas the screenshot above gives an example of a board width that only fits 3 cards per row, the screenshot at the top of this HW specification gives an example of the layout when the board is wide enough to fit 4 cards per row.
* **Note**: If the screen gets wide enough, the last row may have fewer cards than the earlier rows (you do not need to ensure all rows have the same number of cards in this case).

##### Cards

* Each card should be a div element that is 220px wide and 95px tall with a solid black border of 0.35em width and a border radius of 1em.
* Cards will have 1, 2, or 3 shape images (multiple images in the same card must be identical).
* The images in the card should be centered vertically on the card with their distribution across the horizontal axis set to space-evenly.
* Each shape image should have a height of 85% relative to the card.
* When a user hovers over a card, the mouse cursor should change to a pointer icon (hint: there is a single CSS property to accomplish this).
* When a card is considered "selected" during a game (see details in next section), it should have a #636363 shadow shifted 6px right and 6px down:

A picture containing icon

Description automatically generated

%%%%%%%%%%%%%% Above is the required portion for this submission %%%%%

**Behavioral Details**:

Staring a game:

Whenever the "Start" button is clicked, a new game will be started and:

* The current set count should be 0.
* A timer should be started depending on the currently-selected option in the dropdown. If "Unlimited" is selected, the game timer should start at 0 seconds and increment each second. Otherwise, the game timer should start with the number of seconds determined by the selected option and decrement each second (never going below 0 seconds).
* The Main Menu view should be hidden.
* The Game View should be displayed (after the first two steps so that the initial game information is shown correctly when the view changes).
* A new randomly-generated board of cards is generated, where each of the 4 attributes of a card are randomly-chosen (recall each attribute has 3 options) in Standard difficulty. The only difference in the randomness of cards in Easy difficulty is that the "style" attribute should always be fixed to "solid".
* The generated board should never have duplicate cards (cards that share all 4 attributes). During this step, you should not create more than 12 card DOM elements.
  + *Hint for card generation*: Since the provided card images only represent 3 attributes (style, shape, and color), you’ll want to make sure you add COUNT occurrences of the appropriate *STYLE-SHAPE-COLOR.png* image to the card div.
  + *Hint for keeping track of unique cards*: you can give your cards an ID with the 4 attributes in the form: *STYLE-SHAPE-COLOR-COUNT*. Focus on generating these attribute combinations one at a time, creating a new card only when there is not a card already on the board with the same attribute string). But make sure you never have duplicate IDs between the cards on the board.

**Note about difficulty and generated cards**: In both Easy (9 card) and Standard (12 card) difficulties, there are 4 attributes (style, color, shape, and count) and 3 possible options for any attribute (e.g. red, green, or purple options for the color attribute). For Standard difficulty, each card should have a randomly-selected value for each of the 4 attributes. For Easy difficulty, the "style" attribute should be fixed to "solid". To avoid redundancy in your program and allow for flexibility if users later want to change the images and attributes, we suggest using 4 arrays as module-global constants, one for each attribute type (style, color, shape, count) - since count is an integer, you may alternatively use a different datatype to represent it as a constant. Remember that a module-global symbol is any that is in the module pattern but defined outside of the local scope of functions within the module pattern (in general, you should limit the number of module-global variables).

**Note about the game timer format**: When updating during the Game View, the timer format in #time must stay in proper MM:SS format (for example, if there are 8 seconds left in timed mode, the timer should display 00:08. If there are instead 600 seconds, the timer should display 10:00). You do not need to handle the case when 60 or more minutes have passed (which overflows this format).

**How to develop the game?**

Since the board of cards will be populated using JavaScript after switching views, there are a few different strategies you can take for getting started. When starting your JavaScript, you may a) find it easier to implement the behavior for switching between Menu View and Game View, or b) easier to focus on populating the board with cards first (which will require adding CSS properties for card styles). If you’re unsure where to start, the following strategies suggest an approach for implementing the Game View after writing the CSS for Menu View.

1. Write the CSS for the Menu View.
2. Write the initial CSS for the Game View (the details bar and the board border). Hint: Use CSS to hide the #menu-view and show the #game-view in this step while you’re working on the Game View appearance.
3. Write a JS function to create the following Set card and append it to the board in the Game View (containing a single image, striped-oval-red.png):

Icon

Description automatically generated

a) What HTML elements might you identify in this card? The card DOM element you create in the JS function should be built using HTML DOM elements.  
b) Once you have the DOM object created for the card, write CSS (in your set.css file) to apply styles to this card.  
c) Now try adding 3 of the oval images in the card (using JS). Does your CSS need to change to get the following?

A picture containing icon

Description automatically generated

Once you have a function that creates the card DOM object and appends it to the game board in Game View and you have CSS to style the card as specified, extend your JS to append 12 cards on the board. Once you have 12 card objects within the board, move on to Step 4.

1. Implement CSS to achieve card layout on the board as specified in Appearance for Game View.
2. Modify your JS to randomly-generate the 12 cards on the board (**remember you must not have any card on the board sharing 4 attribute values**). During this step, you should not create more than 12 card DOM elements (hint: you can give your cards an ID with the 4 attributes in the form: \*STYLE-SHAPE-COLOR-COUNT\*. Focus on generating these attribute combinations one at a time, creating a new card only when there’s not a card already on the board with the same attribute string).
3. Finish JS to initialize a game starting at the Menu View, using the default options for the difficulty and timer mode.
4. Implement Easy Difficulty (9 cards on the board, all of which have the "solid" style)
5. Implement the game timer requirement for timed and untimed modes.
6. Implement card-selection and Set-checking features (remember to add CSS for selected cards).
7. Implement "Refresh Board" feature (hint: use the same functionality you use when generating the board initially).
8. Test your code for different edge cases, review the spec, and have fun!

Here are some screen shots of the user interactions.

Step 1:

Graphical user interface, application, Teams

Description automatically generated

Step 2: After the User click Start.

Graphical user interface, application

Description automatically generated

Step 3: A set was found (user clicks the individual card and then the boarder is shadowed). Graphical user interface

Description automatically generated

Step 4: More set was found (once the card was used, they can’t be used again for another set).

Graphical user interface

Description automatically generated

Step 5: Time is up. Game is over. Time is returned to 0.

Graphical user interface

Description automatically generated

Tell the user how many sets they found.