Memory Game

CS5003 - Practical 1

Student Id – 200028225

Date – 19th February 2021

Word count: 1440 words

Overview

The assigned project aimed to implement the single-page client-side web application (single-player pair matching memory game) using API. My submission of the project has an implementation of all basic and intermediate requirements along with one advanced requirement (letting the player select the multiple dog breeds) fully and one more at some extent (implementation of modules, closure, and/or classes). The project has one additional functionally to view all game scores with the click of a button.

Technologies and Resources

The project has used HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript technologies to implement the client-side application. The HTML and CSS have been used to design the front-end of the application and JavaScript has been used to implement the backend (at some sort) of an application. The only external library which was being used is Bootstrap CSS to design the application.[2] However, the external CSS (.css) file has been created to customize the styling. Additionally, The W3Schools website has been used to get the idea (how a function works and to gain the knowledge about which parameters can be passed) of window object methods [1]. Other than that, the lecture materials and examples provided by the module instructor have been referred to while achieving the solution of the project. The 'Google Chrome' has been used to run the solution of the project. However, the testing of the project solution has been done on two brewers which are 'Google Chrome' and 'Safari'.

Design

As displayed in figure 1, the player can select multiple breeds and difficulty levels to start the game.

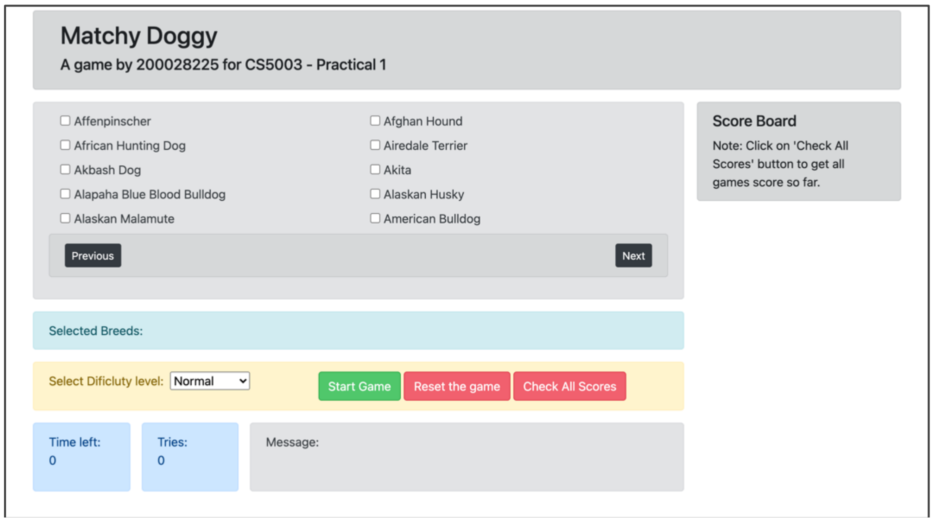
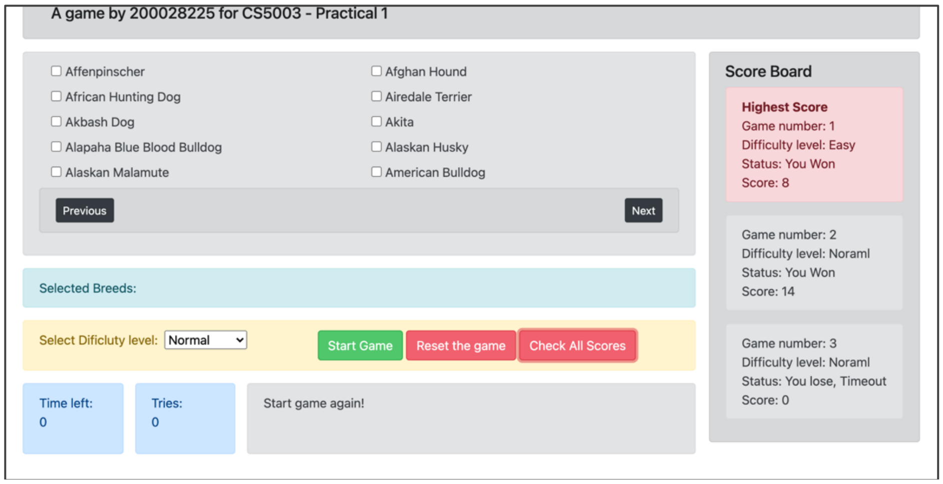


Figure1: The start screen of the developed memory game

The game can be played multiple times with different settings. All scores of the played games will be stored and can be viewed by clicking on the ‘Check All Scores’ button (Refer to figure 2).



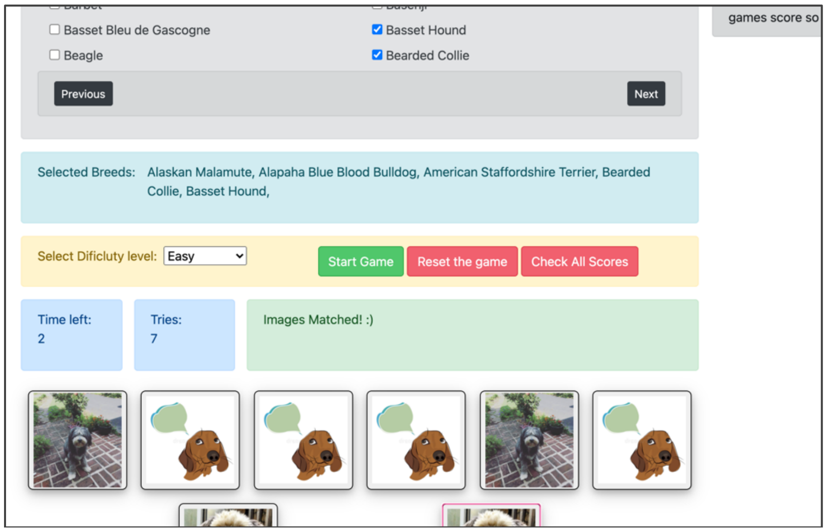
Highest Score
Game number: 1
Difficulty level: Easy
Status: You Won
Score: 8

Game number: 2
Difficulty level: Noraml
Status: You Won
Score: 14

Game number: 3
Difficulty level: Noraml
Status: You lose, Timeout
Score: 0

Figure2: The screen after playing 3 games with different settings

As viewed in figure 3, the checked breeds by a player can be view in the ‘Selected breeds’ section. The time limit and a number of tries can be viewed in the section right above the area of the images. The message of ‘matched’ or ‘not matched’ of player clicked images can also be seen right above the image area. It became easy for a player to keep track of all details for curranty playing game.



☐ Basset Bleu de Gascogne ☑ Basset Hound
☐ Beagle ☑ Bearded Collie

Selected Breeds: Alaskan Malamute, Alapaha Blue Blood Bulldog, American Staffordshire Terrier, Bearded Collie, Basset Hound,

Select Dificluty level: Easy

Time left: 2

Tries: 7

Images Matched! :)

Figure3: The view of screen once player starts the game

The ‘reset the game’ button will pop up the confirmation box to give a player warning regarding the page reloading. The ‘Check All Scores’ button will show a warning message if a player hasn’t played a single game. In both case of winning/losing a game, the system will display the appropriate messages.

The JSON objects, lists, and arrays data structures have been used to implement the solution. On click of a start button, based on the player selection, the API call will be made, objects will get stored into a JSON object. If the player hasn’t selected any breeds or/and difficulty level, the API call will fetch the ‘normal’ difficulty level (6 images) of random images. The JSON object of fetched images will get concatenate with itself to have the 2 same image objects in a list. The elements within the list will get shuffle and the final shuffle object will get a pass as parament to displayImages() and onClickImge() functions.

If a player has selected some of the breeds or/and difficulty level, the solution will fetch all image objects of selected breeds. If API does not have enough images, then it will show an alert message. If API has enough images, the combined list of all selected breeds objects will be created. The random indexes of a difficulty levels will be generated, and objects will be separated of those indexes from the combined list. The image object list will get concatenate with itself and get shuffle and will get a pass as parament to displayImages() and onClickImge() functions to start the game.

Explanation of functions

- `getDogImagesAPI():` This function will try to make an API call and fetch the data. A successful API call with start the game based on played selection and an unsuccessful API call will throw an error message.

- `onClickImge(json):` The function will set the time limit and display it on screen for the player to keep track of time. Each image has an 'event listener' which will notice the players click on 2 images at a time and will do a comparison of images URL in order to display the correct message (images matched or not matched). Each successful matching pair will update the 'win' counter variable and when all matching pair will get found it will clear the interval(timer) and displays the gave over dialog box with winning message along with scores. The function will update the number of tries on the screen too.

- `displayImages(jsonObj):` This function will create the div for images and place the default image of cards in each button element to display on the screen at the beginning of the game.

- `getDifficultyLevel():` The function has switch cases to return the number of images based on the difficulty level. The use of switch case made compilation faster than if-else's statements. This function will get called into getDogImagesAPI() function to only fetch the required number of images.

- `getAllSelectedBreeds():` This function will store all breeds ids checked by the player into a list. This function will get called into getDogImagesAPI() function to fetch the selected breeds from the API.

- `gameoverBox(tries, result) :` This function will store the game number, scores, difficulty level and the result of a game regardless of player won or its timeout situation. It will also display the winning/timeout dialog when a game is over.

- `reset():` The function will get called in gameoverBox() to clear the time limit, tries, breeds selection, and images area of the screen to let the player start a new game.

- `setTime():` The function has to switch cases to return the time limit based on the difficulty level. The function will get onClickImge(JSON) to set the timer when the game starts.

- `getDogBreedsAPI(page):` The function will make an API call to get the list of dog breeds limit of 10 per page.

- `displyBreeds(dogBreedsJson):` This function will create the checkbox and its label and display it on the screen in order. This function will get called into getDogBreedsAPI() function to display the breeds list.

- `pagination():` This function will call getDogBreedsAPI(page) function and pass the parament(page number) "0" , to get the first 10 breeds when page loads.

- `next.onclick:` This function will get triggered with the click of the 'next' button. It will update the page number by '1'and pass it in getDogBreedsAPI() function to display the next 10 breeds from the list.

- `previous.onclick:` This function will get triggered on a click of the 'previous' button to let the player access the previously listed dog breeds by decreasing the page number and passing it into the getDogBreedsAPI() function.

- `breeds.onchange:` Each time player checked the breeds from the list, this function will get triggered and will display the label of the checkbox into the 'selected breeds' area to let the player keep track of selected breeds from different pages.

- `getMinScore():` This function will return the minimum number of tries(best score) from all winning games which were being played by the player without reloading the page.

- `getAllScores():` This function will display the data for 'score board' . It also highlights the best score of the player.

- `reload():` This function will get called when a player clicks on the 'Reset the game' button. It will pop up the confirmation dialog box with waring of reloading the page, the 'OK' button of it will reload the page, the 'Cancel' button will close the dialog box and let a player continue.

Evaluation

As the final solution, the application successfully functions and does most of it which require in a single-page game application. However, the loading of the images can be improved by fetching the images first and loading them on a screen when requires. Currently, a solution does take a bit of time in loading images. It suggested to a player that wait for an image to load after clicking on one image, and then only click on the second image and let the system show you the correct message (images match/not matched). Along with that, the scoreboard does not highlight the scores based on difficulty level, which would have been quite useful for the user. The player has to keep a track of selected breeds via the 'selected breeds' area because the breeds won't be checked again when a player clicks on the next page and again come back to the previous page. However, the game will start with all selected breeds from all different pages, as both tasks are being done by different functions in different ways. This everything could have been achieved if I had more time. The flow of the game can be viewed by uncommenting the 'console.log()' statements from the submission JavaScript file.

Submission folder includes

- Html file (index3.html)
- CSS file (indexCSS.css)
- JavaScript file (indexjs.js)
- JPG file (default dog image - dogthinking.jpg)
- PDF file (report.pdf)

References

1. https://www.w3schools.com/jsref/obj_window.asp
2. https://getbootstrap.com/docs/5.0/getting-started/introduction/