

# CS5003 Master Programming Project Practical 1

Student ID:210003557

Word Count: 1065

## Overview

In this practical, I have accomplished all the basic, intermediate, and advanced requirements.

### Basic Requirements:

- Original layout: When the player hits the start button, the game starts and the start button is disabled. It generates Hearts suit deck (from Ace up to King) and players should choose higher or lower to play the game.

## Higher or Lower

Start

Quit

Choose which deck you want to play with:

☐ Full Deck

☒ Heart Deck



Score:0

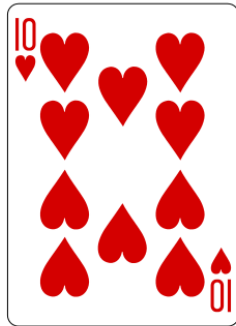
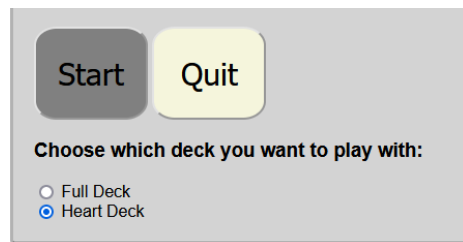
Higher

Lower

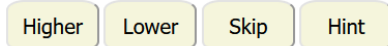
Skip

Hint

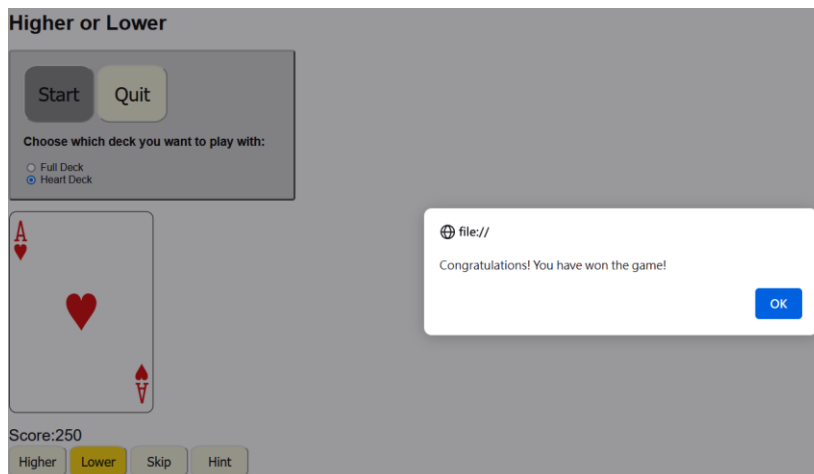
- Correct Guess: The player can keep playing and the score will be added.



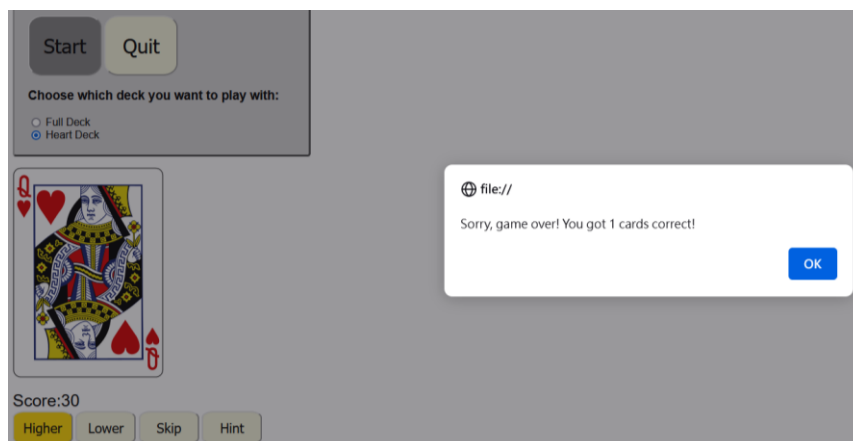
Score:10



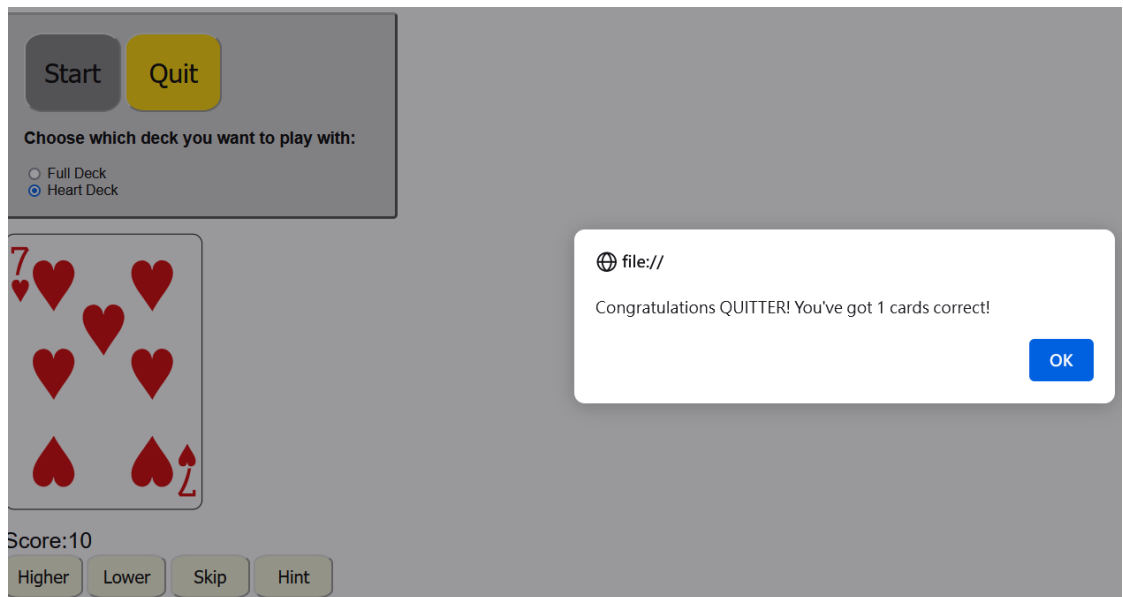
- Win: A pop up message informing players they have won the game.



- Lose: A pop up message informing players they have lost and how many correct guesses has made. The game then ends immediately.



- Quit: The player can choose to quit the game anytime. The message will tell the player has quitted and how many correct guesses were made.



- Track Scores & Correct Guesses: Shown in the pictures above.

Intermediate Requirements:

- Restart without refreshing: After win, lose, or quit, simply re-press the start button and the scores, correct guesses will be re-counted again.

## Higher or Lower

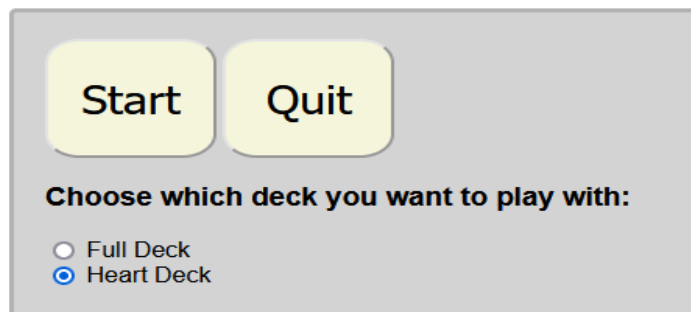


- Track and display 'best' score: For each deck, the highest score, guesses and the time it is played will be written in this board on the webpage.

Deck	Score	Guesses	Date
Full Deck			
Hearts Deck	70	3	2022-02-11 16:55:29

- Choose Deck: Before the game starts, the player can choose either play with full deck or hearts-only deck.

## Higher or Lower



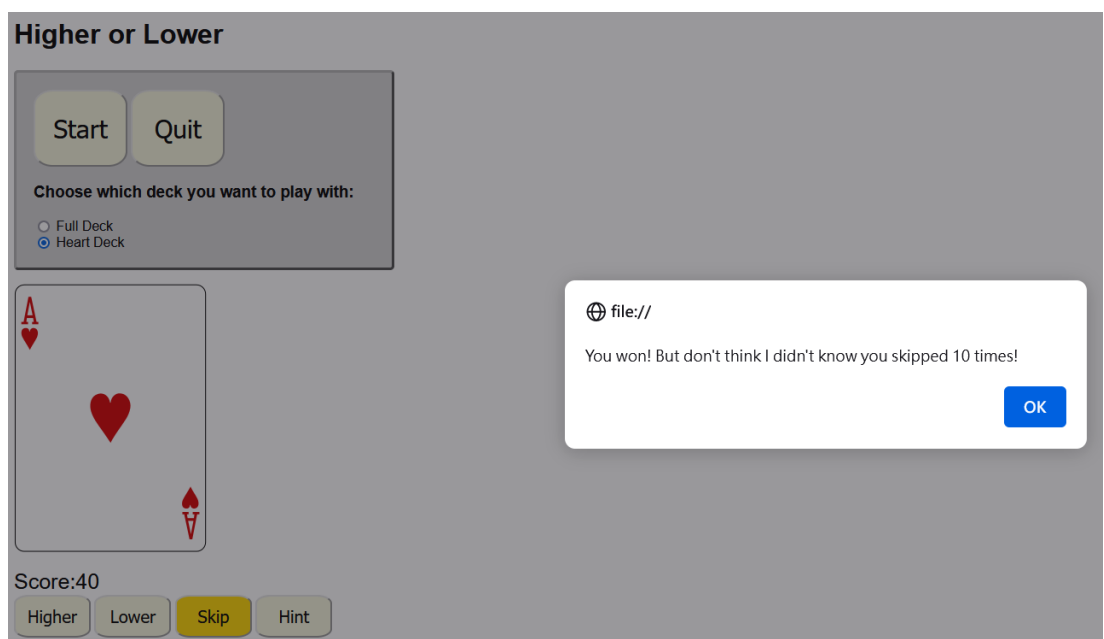
Start Quit

Choose which deck you want to play with:

☐ Full Deck

☒ Heart Deck

- Skip Button: The player can always skip a card and the program will overlook the current card and display the next one. However, the program also counts skip time and will inform the player when they win.



Higher or Lower

Start Quit

Choose which deck you want to play with:

☐ Full Deck

☒ Heart Deck

Ace of Hearts

Score:40

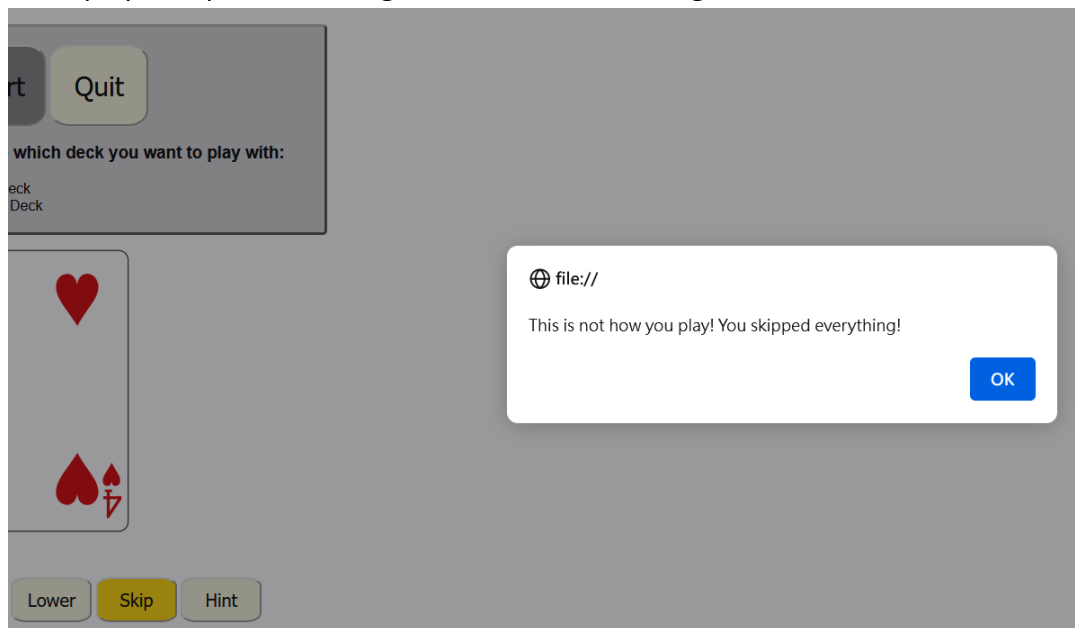
Higher Lower Skip Hint

file://

You won! But don't think I didn't know you skipped 10 times!

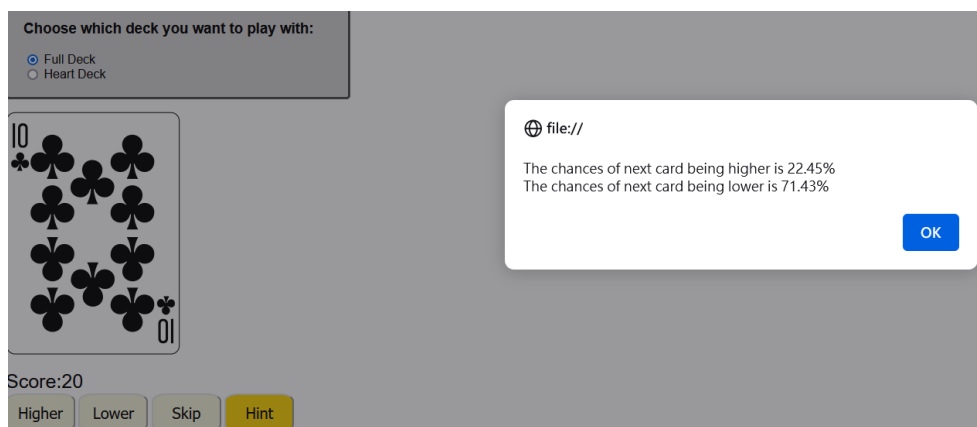
OK

If the player skips the whole game, a different message will show:

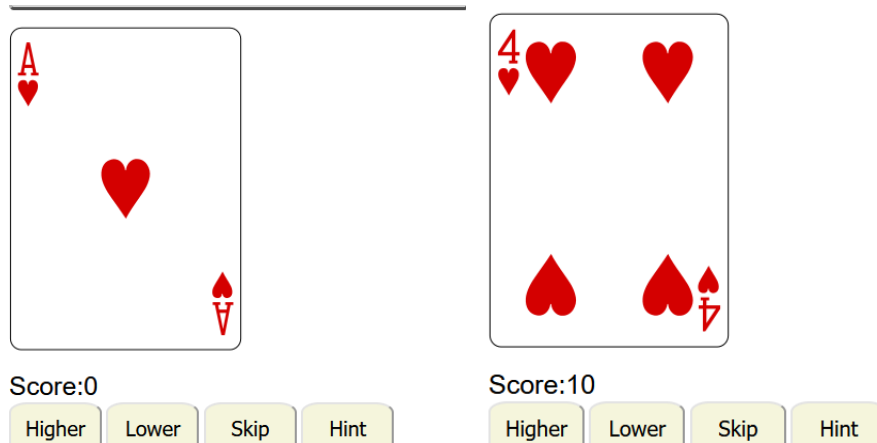


Advanced Requirements:

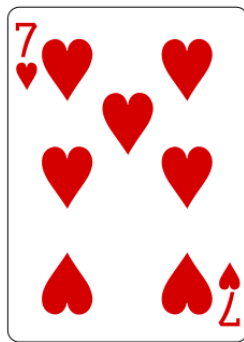
- Hint Button: The player can always press the hint button when they are not sure, it tells them the possibility of the next card being higher or lower.



- Score Scheme: Each card has different scores depends on the difficulty of getting it right. The harder the guess, the higher the score. For example, Ace of heart is an easy guess hence it only gives 10 points.



However, 7 of heart is relatively hard hence it gives a higher score.



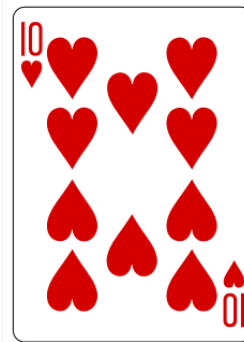
Score:10

Higher

Lower

Skip

Hint



Score:40

Higher

Lower

Skip

Hint

- Local storage: If there is a new record, it will be updated on the scoreboard and will not disappear even if the player refreshes the page. The example below already has a record:

Deck	Score	Guesses	Date
Full Deck	620	28	2022-02-11 19:31:36
Hearts Deck	250	12	2022-02-11 19:30:29

If the players refresh the page or restart the game, the record is still on the board.

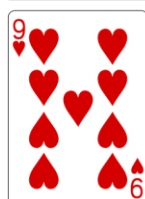
Higher or Lower

Start

Quit

Choose which deck you want to play with:

☒ Full Deck  
☐ Heart Deck



Score:0

Higher

Lower

Skip

Hint

Deck	Score	Guesses	Date
Full Deck	620	28	2022-02-11 19:31:36
Hearts Deck	250	12	2022-02-11 19:30:29

## Technologies & Resources

In this practical, I used some external resources to help me build a more robust program with the right syntax. Since I am still quite inexperienced in coding, there are still some syntaxes in JavaScript that I am not familiar with, the information I searched for was merely to help me use the right syntax in my codes, all the functions and logic were designed by myself. Most of the resources I found were on MDN Web Docs, W3Schools, YouTube, and of course, the Deck of Cards API.

One syntax I spent the most time on was the `async/await` function, which I did a lot of research on all the resources I mentioned above and still was not really familiar with. Most of the difficulties I faced were the extension usage of `async/await` hence it is something I really want to work on myself in the future. The rest problems I had were solved relatively fast thanks to W3Schools and MDN Web Docs.

All the test solutions I ran were on Firefox Browser and the version is 97.0 (64 bits).

## Design

As shown in my codes, I created a lot of functions. The reason was I found these functions and methods quite useful because they not only offered more robustness to my program, but also making them harder for others to change. Also, some buttons in the game were supposed to have the same functions but with different parameters, the functions became handy under this circumstance.

I also utilised a lot of `async/await` functions since I found they would make my codes look neater. A chained `fetch` API could make coding easier for me as I did not need to think about asynchronous problems, but it would also increase the complexity of the codes and make errors occur more easily. Hence, I decided to use `async/await` functions.

Some CSS designs were implemented in my program for specific reasons, I would especially like to bring up the start button. In the demonstration video, the quit button and start button were the same one and would alternate between each other. However, I decided to separate the two because sometimes the players might not notice the change and would keep refreshing the page. Also, it allowed me to implement the `restart()` function which I found quite useful for this game. I also changed the color of start button once the game started to hint the players that the button was no longer clickable until they finished or decided to quit the game, which

increased intuition of the buttons to the users. I also decided to use start button to start the game instead of `window.onload()` because it allowed the users to choose the deck they wanted to play with before seeing the first card. Although I did embed some functions into `window.onload()`, the start button should still be what really started the whole program.

## Evaluation

Overall, I would say this was a challenging but interesting assignment because it challenged my logics as I needed to think about what the outcome would be under different scenarios, especially when math was involved, everything became more complicated. I enjoyed it a lot because I could think about 'how to make things work' with my own logics instead of trying to 'make the website look better'. Although CSS is an important aspect of web design, I personally prefer working on the practical aspect of a program, which is to design a lot of functions and objects then try to put them together to create a complete project that can actually do something.

As mentioned above, the thing I struggled a lot in this assignment was how to correctly use `async/await`, which I spent a lot of time on and still felt somewhat unfamiliar with. I really appreciate this opportunity that I could identify my weakness and confusion in JavaScript so I can improve and research further in the future.