CS 104 Project

Trasula Umesh Karthikeya

June 14, 2023

Contents

1 Introduction **2**

2 Background **2**

3 Game Design **2**
3.1 Main Page 2
3.2 Singleplayer Page 2
3.3 Multiplayer Page 3
3.4 OUTs and Powerups 4

4 Implementation **4**
4.1 Single player 4
4.2 Multi-Player 5

1 Introduction

My project was about creating an online Cricket Cum Minesweeper game. . . I developed a few HTML pages to make this possible. . . The main results were . . .

2 Background

I went through some online code to understand how a Minesweeper game works, CSS styles, JS code format, etc.
I made modifications to some of the codes and observed the changes in the output. I also used ChatGPT to debug errors and understand parts of other codes that I found challenging.

3 Game Design

3.1 Main Page

The main page looks like Figure 1.

Figure 1: Main Page

It contains:

- A box to input the size of the grid.
- Checkbox 1: Singleplayer
- Checkbox 2: Multiplayer
- Start Game Button

3.2 Singleplayer Page

The singleplayer page looks like Figure 2.
In the singleplayer game:

- 11 fielders are arranged hidden.
- Clicking on the grids reveals your score on that grid. You can continue the game until you click on a grid with a fielder.
- You may receive power-ups that double your current score if you are lucky.

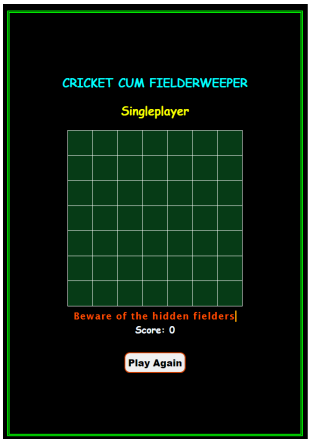
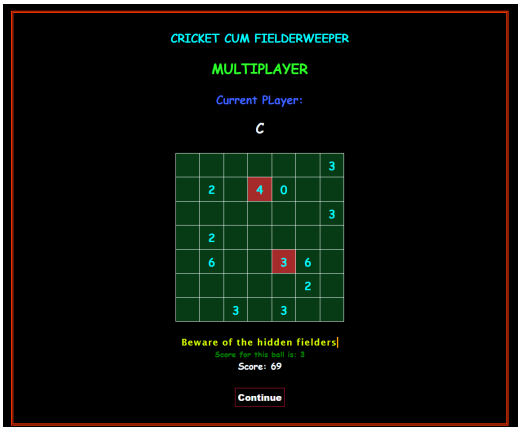


Figure 2: Singleplayer Page

3.3 Multiplayer Page

The multiplayer game consists of two pages shown in Figure 3.

(a) Multiplayer Page 1

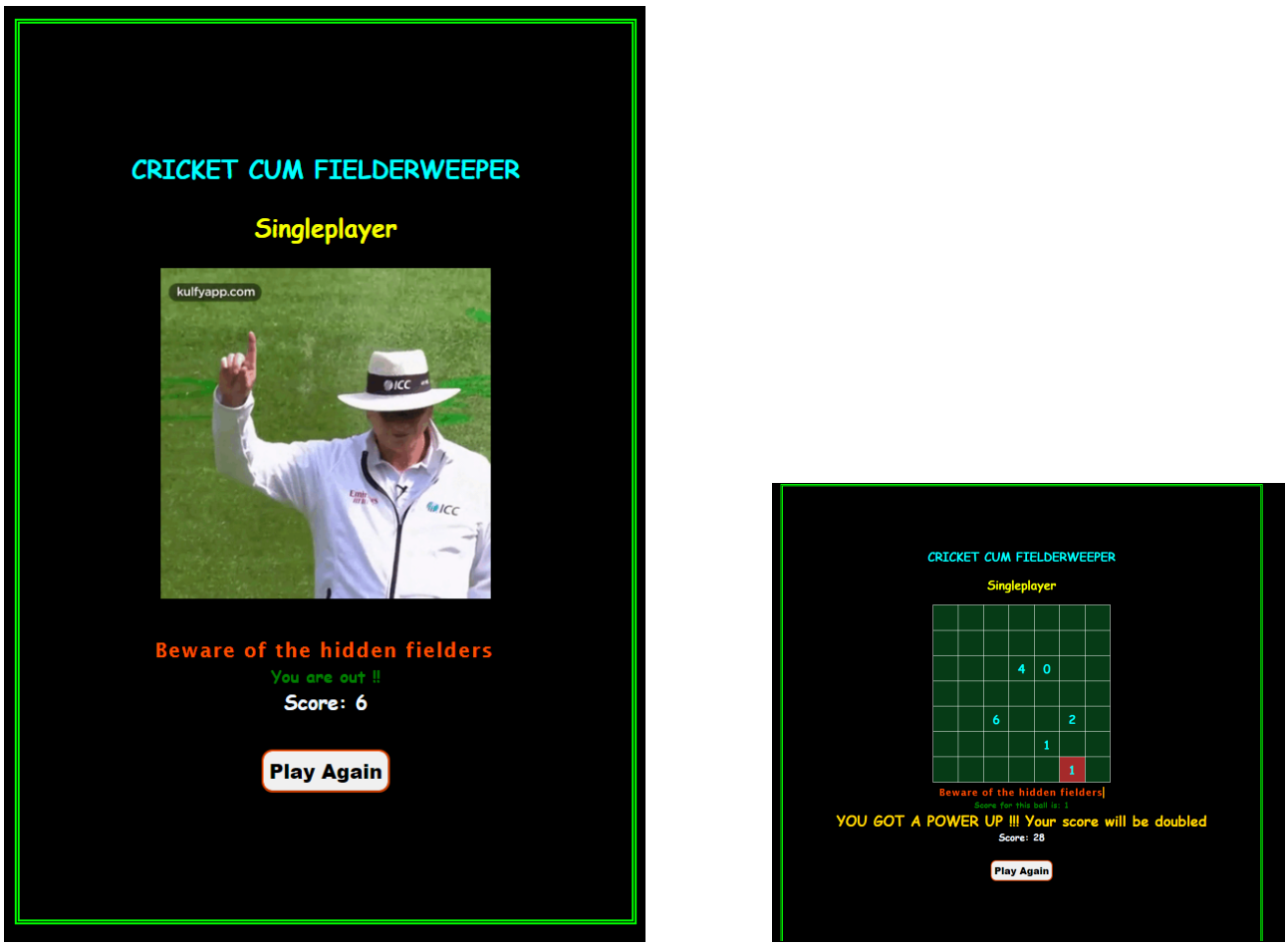


(b) Multiplayer Page 2

(c) Multiplayer Page 3

Figure 3: Multiplayer Pages

3.4 OUTs and Powerups



(a) OUT (b) PowerUP

Figure 4: Multiplayer Pages

4 Implementation

4.1 Single player

In the single player game I have created a grid and arranged 11 players randomly in $n*n$ using *Math.random()* function.

I have used the following functions:

- **handleCellClick()**
  Using this function we can handle any grid using a mouse click. It has two *3 if-else* conditions. One for *if clicked grid has a mine* (Game will be ended by calling *endGame()* function).
  Second one checks weather game is ended or not using a bool variable *gameEnded*.
  Third one is for the powerup. I have created a random number less than $n*n$ and divided it by 10 and then compared the remainder with 7. If this is true then you will get a powerup. I have done this operation to as a random operation as we need to assign a random grid for power up.

And score is also generated in this function score can be from 0 to 6 except 5. I have exculded 5 by using a while function *"while (randomnumber == 5)randomnumber = Math.floor(Math.random() * 7);"*
And there will be variable called *Score* which calculate the final score by adding the individual scores of each grid and also Doubling when it got a powerup. Individual scores of each grid will be visible on that particular grid by using *cell.textContent = randomnumber;* here *randomnumber* is the present score for the ball.

- **endGame()**
  It will assign the *bool gameEnded* as true. It will clear the grid by using
  let cleanet = document.getElementsByClassName("table-container");
  cleanet[0].style.display = "none";
  I have displayed the umpire gif by using
  let out = document.getElementById("fig");
  out.style.display = "block";
  It will have an if cond. containing the variable *endg* (Used as a short cut for endgame) it is number of times you have clicked the grid if *"(n*n - endg)"* is equal to 11 then the player who played the will be declared as the winner in this *Singleplayer Game.*

- **Restart()**
  This function will be actived by clicking **Play Again** button in Singlplayer page.
  This function reset the game by using *location.reload();* which reloads the current web page.

- For more information refer [1], [2]

4.2 Multi-Player

For the implementation of Multiplayer game I have used two html pages *Multiplayer.html* and *supmul.html* (short name as support for Multiplayer).

Multiplayer.html have the following features:

- This page will have two arrays which will be used later one for player names and other for player scores (all the scores were assigned to zero).

- This page will have a box that takes number of players (greater than 1) as input and then their names and stores them in the array playernames.

- Now this page will store:

Variable	**How it is stored in Browser's local storage**
playerscores	localStorage.setItem("playerscores", JSON.stringify(playerscores));
playernames	localStorage.setItem("playernames", JSON.stringify(playernames));
number	localStorage.setItem("number", number);
numplayers	localStorage.setItem("numplayers", numplayers);
numtimes	localStorage.setItem("numtimes", numtimes);

supmul.html will work based on the following functions and strategies:

- First we will use the following to obatin the respective variable:

Variable	How it is stored in Browser's local storage
playerscores	JSON.parse(localStorage.getItem("playerscores"));
playernames	JSON.parse(localStorage.getItem("playernames"));
number	localStorage.getItem("number");
numplayers	localStorage.getItem("numplayers");
numtimes	localStorage.getItem("numtimes");

- My Strategy in this multiplayer is :
  - Calling this page recursively by number of players times
  - Each time when the page is called it updates the array *playerscores* by pusing the score of the each player by using *playerscores.push(score);*.
  - Then when then final player clicks the continue button the grid and some extra elements will be disappeard from the screen. And scoreboard will be visualised.
  - *Scoreboard* will show the players names and their respective scores. It will also show the winner name *Player with maximum score*.
  - If any of the two players have same score then it will show *There is a tie* and the player names.
- Function used in this page are :
  - *handleCellClick(event)*
  - *endGame()*
  - *showscores()*
  - *Restart()*
  - *createagain()*
- function *handleCellClick(event);*
  This is the same function that we have used in the case of **Single-Player**.
- funtion *endgame();*

```
1              gameEnded = true;
2              let cleanet = document.getElementsByClassName("table-
                  container");
3              cleanet[0].style.display = "none";
4              let out = document.getElementById("fig");
5              out.style.display = "block";
6              playerscores.push(score);
7              numtimes++;
```

  The function is same as that in Single-Player but only difference is it also counts numtimes the game played by increasing the variable *numtimes* by one.
- function *Restart()*
  This will have a slighter difference with the function in Single Player.

```
                if (numtimes == numplayers) {
                showscores();
            }
            else {
                localStorage.setItem("playerscores", JSON.
                   stringify(playerscores));
                localStorage.setItem("playernames", JSON.stringify
                   (playernames));
                localStorage.setItem("number", number);
                localStorage.setItem("numplayers", numplayers);
                localStorage.setItem("numtimes", numtimes);
                window.location.href = "supmul.html";
            }
        }
```

This will show the **Score board** by comparing numtimes with numplayers or else it will update the array *playerscores.*

- function *showscores()*
  - Intially this will remove the *grid, type-writer, buttons, etc.* and *Scoreboard, Playernames, Playerscores, winner*
  - We will find the index of maximum score by

```
                  let maxnumber = -Infinity;
            let maindex = -1;
            for (let i = 0; i < playerscores.length; i++) {
                if (playerscores[i] > maxnumber) {
                    maxnumber = playerscores[i];
                    maindex = i;
                }
            }
```

  - Now if any two player score the same maximum score then a tie will be given to them or else the player who scores maximum will be declared as winner.

```
                              let j = 0;
            let equal = [];
            equal.push(maindex);
            for (j; j < playerscores.length; j++) {
                if (playerscores[maindex] == playerscores[j]
                   && j != maindex) {
                    equal.push(j);
                }
            }
            if (equal.length == 1) {
                let winner = playernames[maindex];
                document.getElementById("output3").innerHTML =
                    "The Winner in the game is " + winner;
            }
            else {
                let ties = [];
```

```
                for (let z = 0; z < equal.length; z++) {
                    ties.push(playernames[equal[z]]);
                }
                document.getElementById("output3").innerHTML =
                    "There is tie between " + ties;
            }
        }
```

  - In the above code I have declared a variable j, array equal.
  - Then it will check score of the player with *maindex* with other indices and if there any the index of that score will be noted in array *equal*. And name of the players will be noted in array *ties*.

- Function *createagain()*
  This will create the grid again and this function will be activated when clicked on *Continue* button.
  The code is as follows:

```
                    for (let i = 0; i < numRows; i++) {
            let row = document.createElement('tr');
            grid.appendChild(row);

            for (let j = 0; j < numCols; j++) {
                let cell = document.createElement('td');
                cell.dataset.row = i;
                cell.dataset.col = j;
                cell.addEventListener('click', handleCellClick
                    );
                row.appendChild(cell);
            }
        }

        // Randomly place Fielders on the grid
        for (let i = 0; i < numfielder; i++) {
            let randomRow = Math.floor(Math.random() * numRows
                );
            let randomCol = Math.floor(Math.random() * numCols
                );
            let cell = grid.rows[randomRow].cells[randomCol];
            cell.dataset.hasMine = 'true';
        }
```

- For more information refer [3]

I have used [4] for debugging.

References

[1] URL: https://codepen.io/joelbyrd/pen/YPKPbw.

[2] URL: https://iq.opengenus.org/minesweeper-game-using-js/.

[3] URL: https://www.w3schools.com/html/default.asp/.

[4] URL: https://chat.openai.com/.