Instituto Superior de Engenharia

Politécnico de Coimbra

Soup of letters

Practical Assignment – Script Languages

Michal Czysz – 2021113916 - Erasmus
Pedro Martins – 2021118351 - EI-CE

1. Introduction

During current semester on script languages classes was taught how to deploy basic web applications with JavaScript and it framework - React. To validate the knowledge, We performed this obligatory practical assignment.

2. Description of program

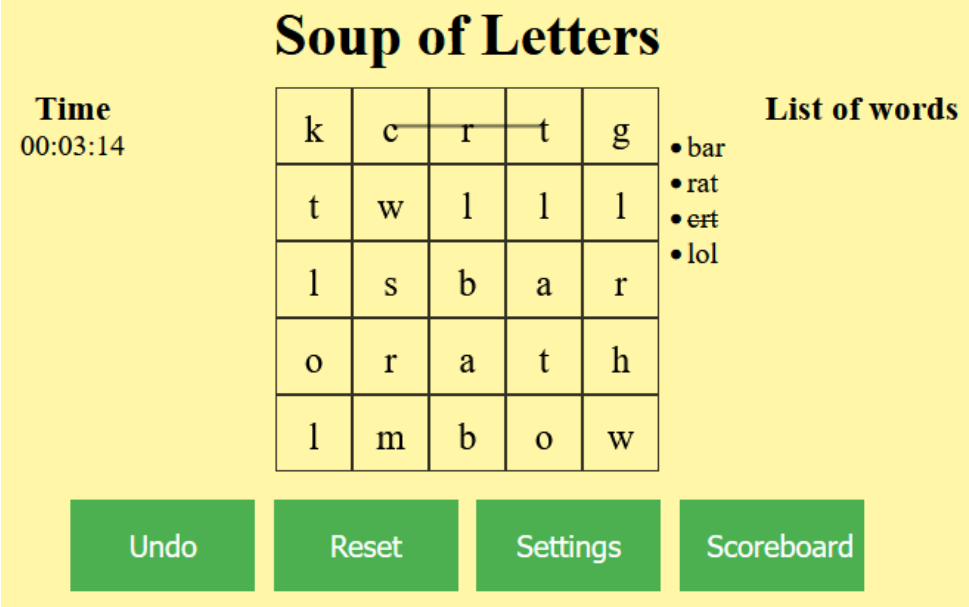
The work was about making Soup Of Letters game. The goal was to write web game that consists of grid of letters and find correct sequence that creates words, among randomly generated letters.

Visualization of game idea

We established few rules for our game to follow them while writing the program:

- Player can “Undo/Back” the move infinite amount of times
- Words can be in vertical, horizontal or diagonal orientation
- Words are always written from left to right
- Player can mark the words from left to right as well as from left to right
- There are three levels of difficultness – 5x5, 10x10, 15x15
- Final score depends on time how fast game was completed, how many words were on board and what level was chosen
- Player can add and delete words from wordbook
- We are not going to use any extra libraries or framework

3. Main interface



Main interface of game

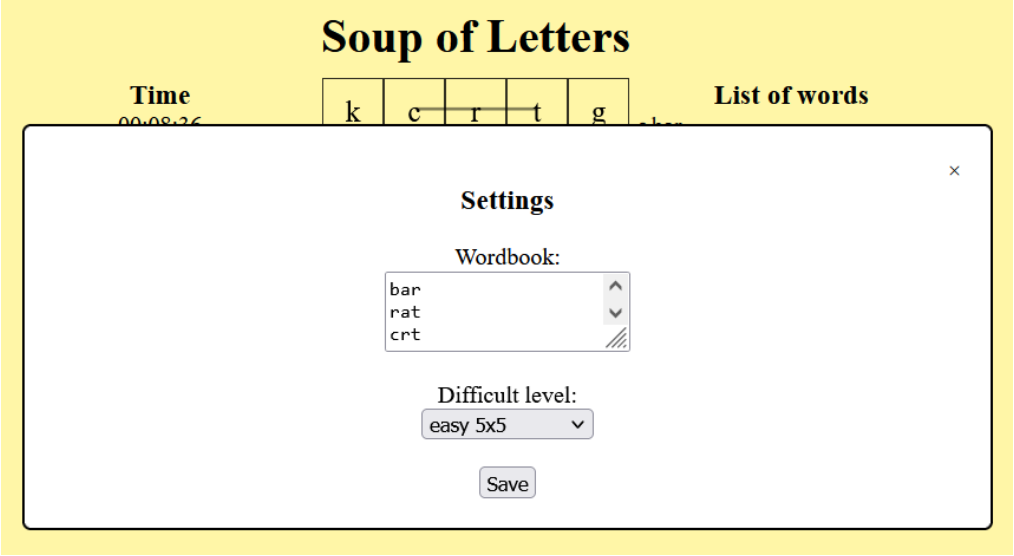
It was decided to stay with simple design, close to the one presented in provided description of practical assignment. On the left side of game board timer was added (for scoring purposes), on the bottom we placed some buttons and on the right side is list of words/wordbook that can indicate when the correct word was crossed.

At the beginning script generates 2D array and reads the words from local storage (player can put own words there), push them to special array where assign to them the direction how they will be placed od the grid and if they will be crossing.
This function in return gives information about used words and their positions which is later "saved" in useState hook.

```
const [grid, setGrid] = React.useState(Game(dim)) //In Game() we generate grid
```

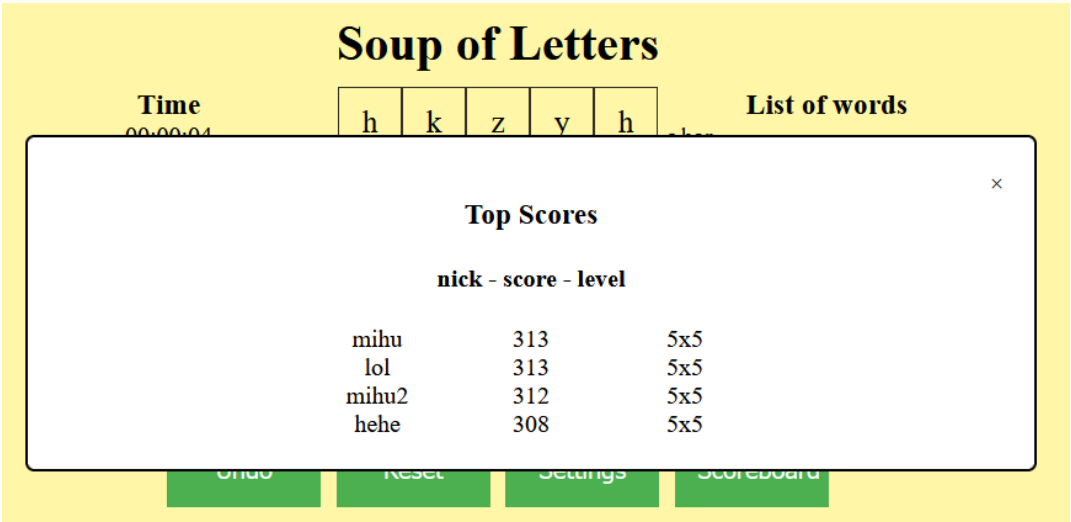
4. Settings and Scoreboard screen

There are two buttons that after clicking on them “pops on” new windows with different options to change or with previous scores.



Settings menu

Settings menu more complex in terms of used JavaScript because here we had to read and write to the local storage all of the settings we change every time.



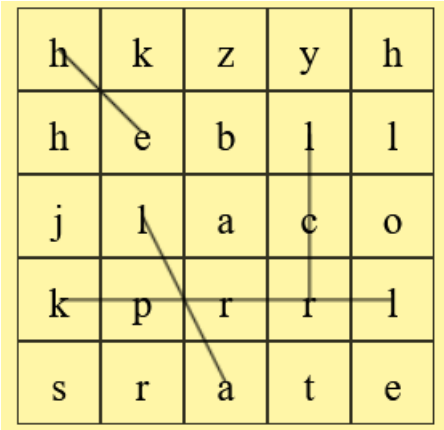
Scoreboard window

Window with scores was more demanding in CSS because every score is in separate grid cell, and it was needed to spend some time to make everything centered and even.

5. Gameplay

The rules of game were already described yet we implemented some extra functions worth to mention.

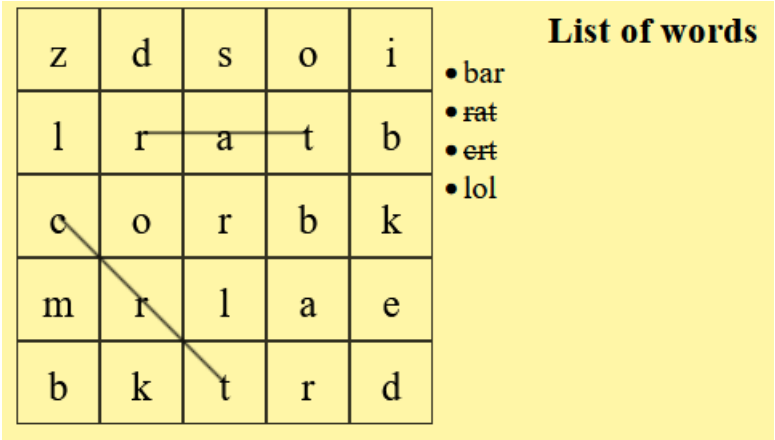
- Drag to select – to select sequence of words we just need to click-down somewhere on the board and start dragging the cursors towards last letter we want to mark. At the same time it will be drawn line on selected letters.



Lines drawn on gameboard

To draw those lines, we used Canvas that is placed under the grid – lower Z-Index. Canvas and it lines could be accessed as well from different function, for example after clicking “Undo” we needed to delete last drawn line so the canvas have its own reference to help deal with that.

- Indicating correct marked word – as its state, when we selected correct sequence of letter and we found the word from word list, that word is check on wordbook list



Check words on wordlist

- Scoring mechanism – we prepared simple formula to score player after completing the game: (250 - TimePlayed + (AmountOfWords * 10) + Level*10)

6. Conclusion and further work

All of the goals provided in practical assignment document were achieved but there are still some things that can be improved. First thing is to try to make the code easier to read, it can be done by moving some of the code from from *App.js* file to some other file and then just to include it. Another minor improvement could be to re-write some parts of code, to make it more consistent – write it in same manner as rest of those checks/if functions for example. We would like to point as well need to improve component file placing and its names, to make them more easy to work in the future and by different persons.