Sudoku Trial Day

In this project, we expect you to implement a Sudoku game with the following requirements & conditions:

Game rules:

- All classical Sudoku rules must be followed
- There will be 4 ranks:
    - beginner (36-40 cells visible out of 81)
    - intermediate (32-36 cells visible out of 81)
    - hard (28-32 cells visible out of 81)
    - expert (24-28 cells visible out of 81)

- Those pre-filled cells won't be editable.
- Any error input will be seen immediately.
- A game must be randomly generated: every new game is different from previous.
- There will be a Records table with the top-3 highest results from all ranks, which will stay the same on page refresh.
- Only successful games are eligible for the Records (Leaders) Board.
- There will be a hint system, which will be shown as a bulb, once the user clicks on the bulb, the hint will be given.
- Using 10 hints at most will be possible.
- The scoring system will be as follows:
    - Part 1:
        - Every correct cell will result in +5 points
        - Every hint will result in -3 points
        - Every next hint will be -1 from the previously subtracted hint points (i.e. if first hint has decreased 3 points, the next will result in minus 4, the next next in minus 5, etc)
        - Every error will result in -1 point.
    - Part 2:
        - After the whole sudoku is done, the time elapsed (in seconds) will be subtracted from the constant value of 500 and added to the total score (i.e. if the user has finished in 120 seconds, the total score will be

```
TOTAL_POINTS_FROM_PART_1 + (500 - 120)
```

- Available digits (for the whole sudoku grid) will be listed and, in case when all possible values are filled, the related digit will be gray.
    - Example: if the user finds all 9s in the whole sudoku, the 9 in the available cells will be grey.

Project:

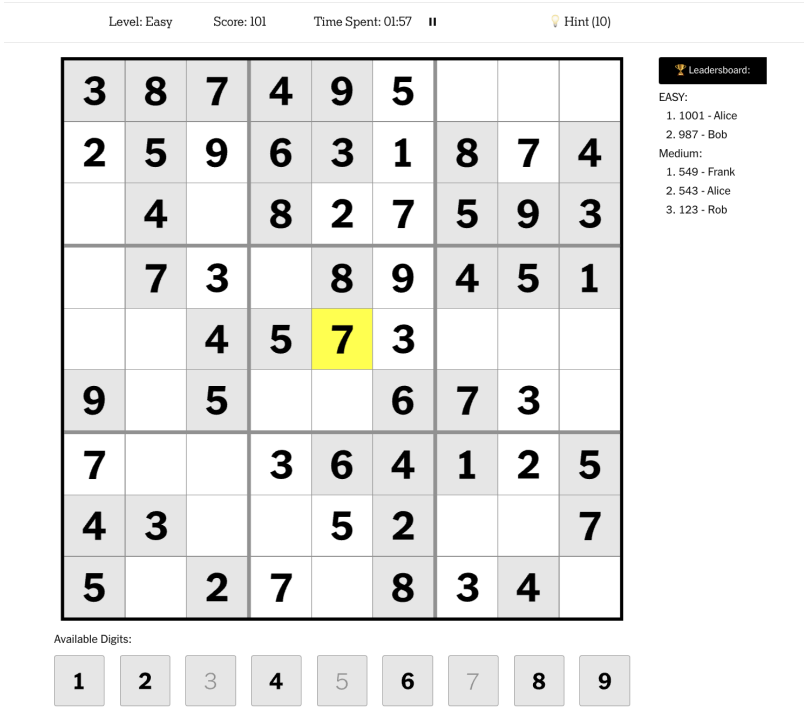
- The user will fill the cells from the keyboard.
- We expect lines or 3x3 cells to animate when all values are found successfully.
- We expect a cell to show the error, when there's one.
- We expect your own code styling just like in a large scale project.
- We expect the project to be done in the Vue.js & TypeScript.
- We expect TypeScript's type definitions with a proper `tsconfig` file.
- We expect unit tests to be written (a must for the algorithm part).
- You can
    - use any compiling tool,
    - use any colors or styling,
    - animate the winning game,
    - ask any question, anytime.

Bonuses:

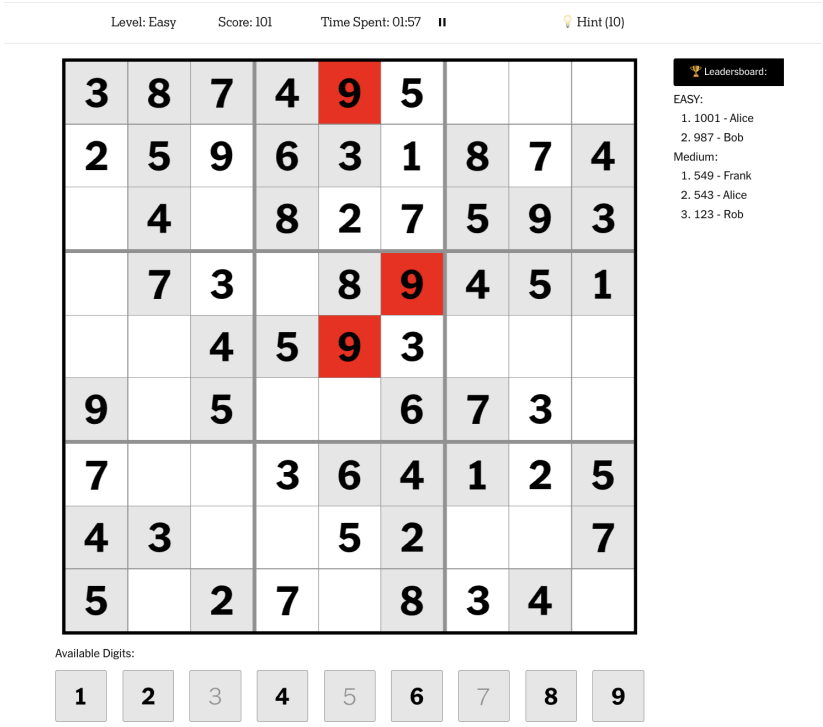
- dockerizing the project,
- adding Undo-Redo button,
- adding draft option (can be italic),
- pausing game or auto-pause on the tab change,
- saving the records to the Database with the back-end implementation in Node.js & TypeScript.

Example project:

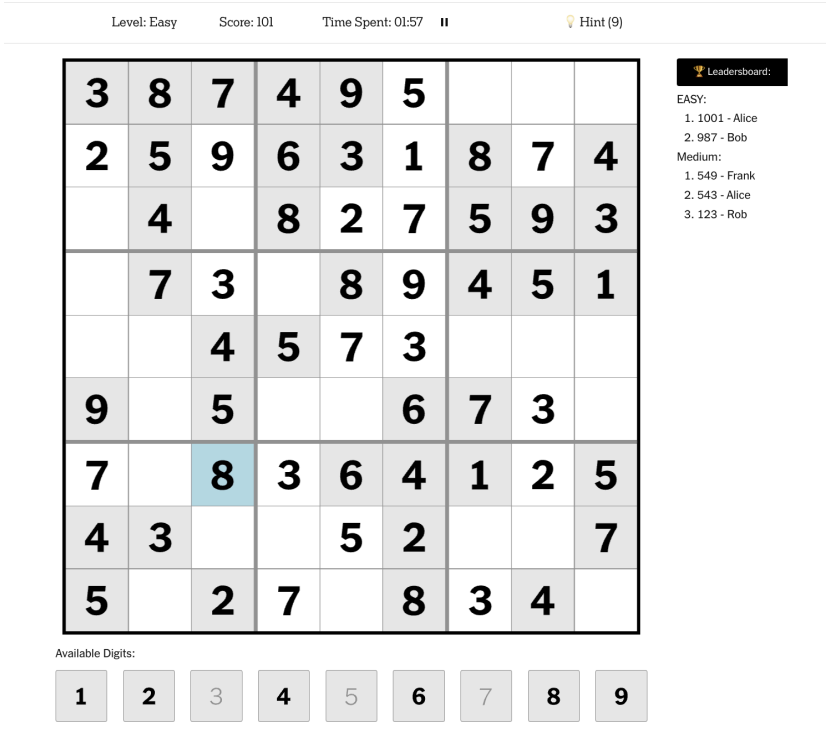
General view and correct input given:



Error input case:



Hint usage:


Level: Easy
Score: 101
Time Spent: 01:57
Hint (9)
3 8 7 4 9 5
2 5 9 6 3 1 8 7 4
4 8 2 7 5 9 3
7 3 8 9 4 5 1
4 5 7 3
9 5 6 7 3
7 8 3 6 4 1 2 5
4 3 5 2 7
5 2 7 8 3 4
Available Digits:
1 2 3 4 5 6 7 8 9
Leadersboard:
EASY:
1. 1001 - Alice
2. 987 - Bob
Medium:
1. 549 - Frank
2. 543 - Alice
3. 123 - Rob