Masai Word game (Full-stack)

Instructions

- Use React JS to solve this question.
- Keep the code clean, commented and documented. Maintain feature based coding. (separate action and reducers - feature wise folders)
- You are free to use any css solutions as long as it looks good. Remember this also has score
- You will have to also deploy both backend and frontend server and app.

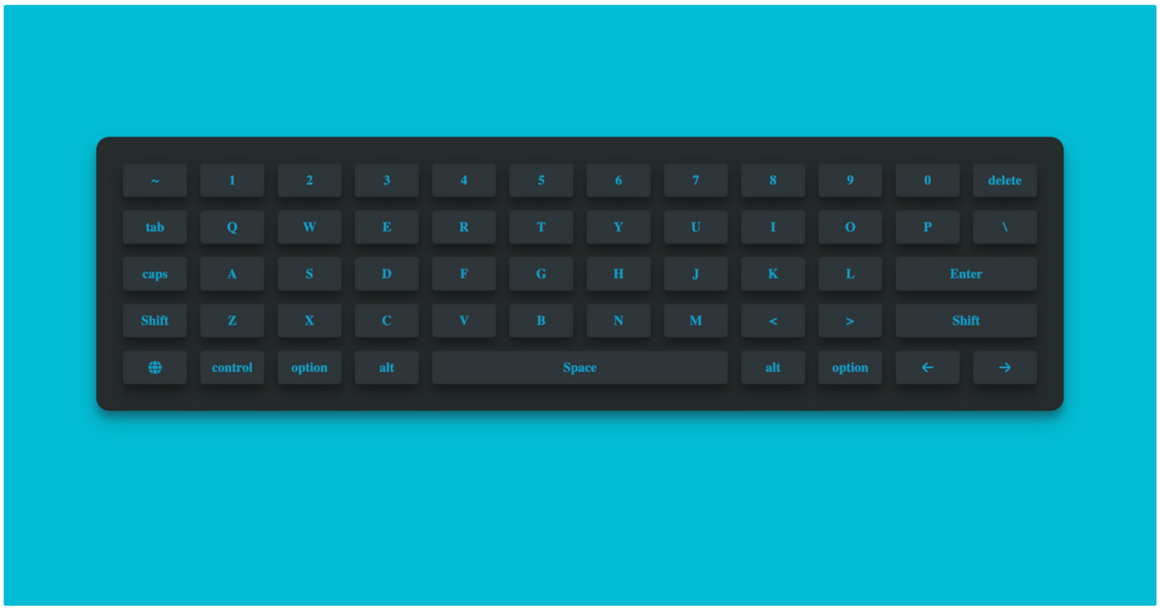
Problem Statement:

- Build a word game where user can play using a virtual keyboard.
- Your home page should have a form which takes following information from the user
    - Name
    - Difficulty level
- On form submit user should be move to `/playzone` route

Playzone page

- Make your own API to generate a random word. (for reference check this API : https://api.api-ninjas.com/v1/randomword)
- Your task is to generate a random word using your API and when user types the same word on virtual keyboard his score should increase.
- This page should have a virtual keyboard, where user can click on any alphabet

- Refer to this UI (display random word on top of this keyboard)


~ 1 2 3 4 5 6 7 8 9 0 delete
tab Q W E R T Y U I O P \
caps A S D F G H J K L Enter
Shift Z X C V B N M < > Shift
control option alt Space alt option ← →

- When user clicks on any key, display those keys on UI (similar to keyboard functionality)
- When length of typed keyword equals length of random word then check whether the typed word is correct or not with random word, also move user to next word simultaneously.
- When user types in correct random word, then score should be increased else decreased.
- Score should be increased/decreased based on random word length
  - If random word length is 4 and user types in correct word through virtual keyboard - score should be 4
  - If next random word length is 6 and user types in correct word through virtual keyboard - score should be 4+6 = 10
  - Similarly for wrong answers as well, but for wrong answers scores should be subtracted.
- This page also has a timer. Time limit for game should be changed according to difficulty level
  - High level - 10 seconds
  - Medium level - 20 seconds
  - Low level - 30 seconds
- Game should end when timer reaches 0 seconds and score should be displayed.
- Store individual player result in backend server along with level, score and name.

Dashboard page

- Display all users score along with names in this page in tabular format.
- Maintain order as highest to lowest in terms of score.

Note:

- Maintain flow of app as mentioned.
- Use components based structure.
- Error messages should be shown, make use of any CSS library of your choice (chakra and MUI preferred)
- Use loaders.
- Good designs will fetch bonus marks.
- Submitting local host links for mock server will lead to disqualification.

Submission

- Please submit deployed link and Github link of code.
- **Push your code into `masai-repo`, don't submit personal repo links (submitting personal repo links will lead to disqualification)**
- Please double check if deployed version works or not (run deployed version on your laptop and then submit it).
- Any issues in deployed link, will be considered null and void.
- Please verify your submissions are correct.
- Make sure you follow all instructions carefully.
- Submit before deadline.

Rubrics / Criteria to be judged upon

- HTML, CSS, React, **Redux**
- Filtering, sorting, pagination.
- Code cleanliness.
- Component structure and **Good Git Hygiene.**

Time Limit - `4 Hours`