Masai Quiz (Full-stack)

Instructions:

- Read each and every line of question carefully, missing any functionality which was asked will reduce your overall marks.
- Use react-router for routing.
- Use redux and thunk for dispatching asynchronous actions.
- Keep the code clean, commented and documented. Maintain feature based coding. (separate action and reducers - feature wise folders) example : all the action and reducer related to authentication in one folder etc.
- You are free to use any css solutions as long as it looks good. Remember this also has score

Problem Statement:

Frontend:

- There should be two buttons which will redirect to
  - Admin page
  - Quiz page

Admin Page:

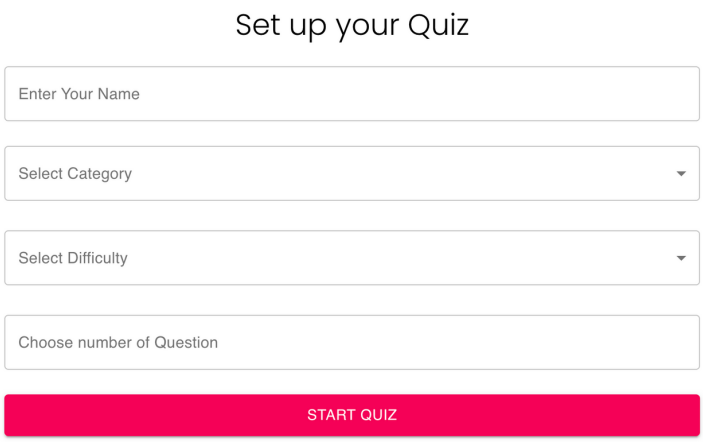
- The route for admin page should be `/admin`.
- Admin should be able to login using https://reqres.in/
- After logging in display token.

- The admin page should contain a form through which the admin can upload questions to the database. Keep all questions to be of multiple choice format for in this application. The form should contain the following fields:
  - Category - select tag with different options such as `Geography`, `Sports`, `Vehicles`, `Music`, etc.
  - Difficulty Level - select tag with options as `Easy`, `Medium` and `Hard`
  - Question - Input box with placeholder text as `Enter question`.
  - Correct Answer - Input box with placeholder text as `Enter correct answer.`
  - Option 1 - Input box with placeholder text as `Option 1`.
  - Option 2 - Input box with placeholder text as `Option 2`.
  - Option 3 - Input box with placeholder text as `Option 3`.
- On form submit store these questions in backend through API.
- Store at-least 10-15 questions.
- Refer to this API to get idea of questions. - https://opentdb.com/api_config.php

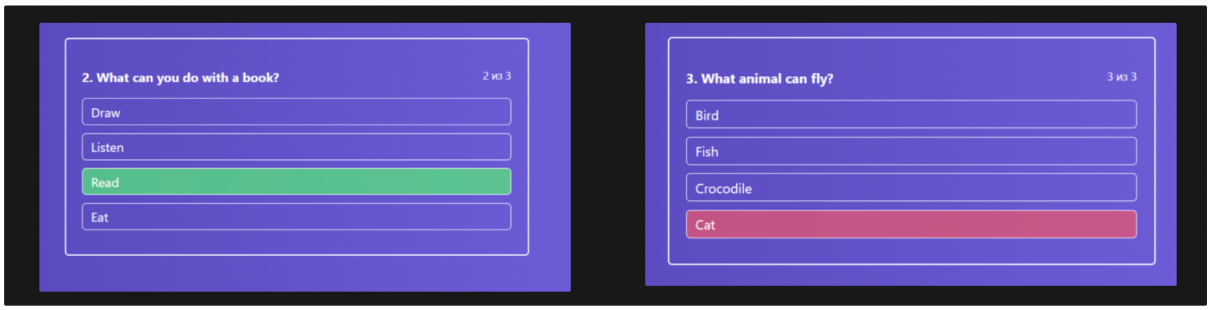
Quiz page

- In this page user can set up quiz with following information
  - Name of the user
  - Category (General Knowledge, Sports, Geography)
  - Difficulty level
  - Number of questions

  Refer to this image below



- When clicking on start, user should go to `/questions` page where questions corresponding to selected category, difficulty level should be fetched.
- UI should look something like this for correct answer and wrong answer (just a reference, you can build your own UI)



- On clicking on any option, user should be able to see if his answer was correct or wrong (you can also use toaster or modal for this)
- There should a Next button and Previous button as well, On clicking on `Next`, user should be moved to next question, On clicking on `Previous`, user should be moved to previous question.
- When user reaches to last question, there should be submit button instead of next, when clicking on submit user should be moved to `/results` page.
- Keep count of correct answers made by the user (you can store count in redux)
- In `/results` page display result in form of tabular format, which should include
  - Correct answers count
  - Incorrect answers count
  - Total score
  - Percentage
- You should have one more route for `/dashboard`.
- In Dashboard page show results of all users who participated in quiz along with their scores.
- User who scores highest marks should be at the top of list.

Backend:

- Build relevant APIs to store questions which admin has posted.

- Build relevant API's to get questions on frontend for different filters
  - Category
  - Difficulty level
  - No of questions
- All the filters should work together.

Submission

- Please submit deployed link and Github link of code.
- 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
- APIs will be tested.
- React router
- Code cleanliness

Heading 3