Your task is to create similar **SPEEDTEST** by using React.js

1. Create new React app in **reactjs_sandbox**
2. Clean your app
3. Update **App.js** to have a class component
    - Add heading
    - A placeholder for score
    - Buttons for start and end game
4. Make overall **CSS** changes
    - Add gradient background
    - Change font (use playful font)
    - Style also buttons
5. Create a new separated function component: **Circle.js,** and connect it with App.js
    - Make circles using Circle.css
    - Show min 4 circles on the application (use a map())
6. Make **circles clickable** and **update the score** by state
    - State -> score: 0
    - clickHandler -> setState - this.state.score +1
    - connect clickHandler with circles -> {props.clicks} and {this.clickHandler}
    - pass circle number to clickHandler -> use data passing to the event handler (binding)
    - show updated score in score placeholder -> see step 3.
7. **Finding a random number** for a random circle highlight
    - You need a random number from 1-4 (or how many circles you have) -> https://www.w3schools.com/js/js_random.asp
    - Add state -> current: 0
    - Use the ***Do While*** **loop** to find a number which is 1-4 but not the same as it was previously
    - Use setState for that number generated randomly -> setState - current : nextActive
8. **Add a timer for random numbers** (use, for example setTimeOut method)
    - Define speed and timer
    - add start handler
    - add end handler -> use clearTimeOut
9. **Colours changes** (use inline styling and conditional rendering)
    - Add **different colours for all circles** -> check is circle default or active (the random number we created) and use active colour or inline style background colour
    - Add **highlight colour for active circle** -> add class in CSS for the active state (for example, default is „.circle“, but in the active phase, it is „.circle .active“)
10. Create a **GameOver.js** function component and connect it with App.js
    - Create **overlay**
    - Create a **popup box**
        - Add Heading
        - Add Score
        - Add close button
    - **Style** GameOver view
11. By using a true/false state, **hide the GameOver component** until endHandler is triggered.
    - State -> showGameOver: false

- Add trigger -> endHandler setState - showGameOver:true
- Wrap component in JavaScript code which checks is state true or false

12. Add the **right circle, click check**

- If the randomly generated number and circle ID does not match, then endHandler will be triggered

13. Add **rounds to end the game after five rounds** (if the player does not click five rounds, then the game will end)

- State -> rounds: 0
- setState – this.state.rounds + 1
- Add round check -> if more than five, then endHandler will be triggered
- Add in clickHandler setState, which will clear rounds if the user clicks circles.

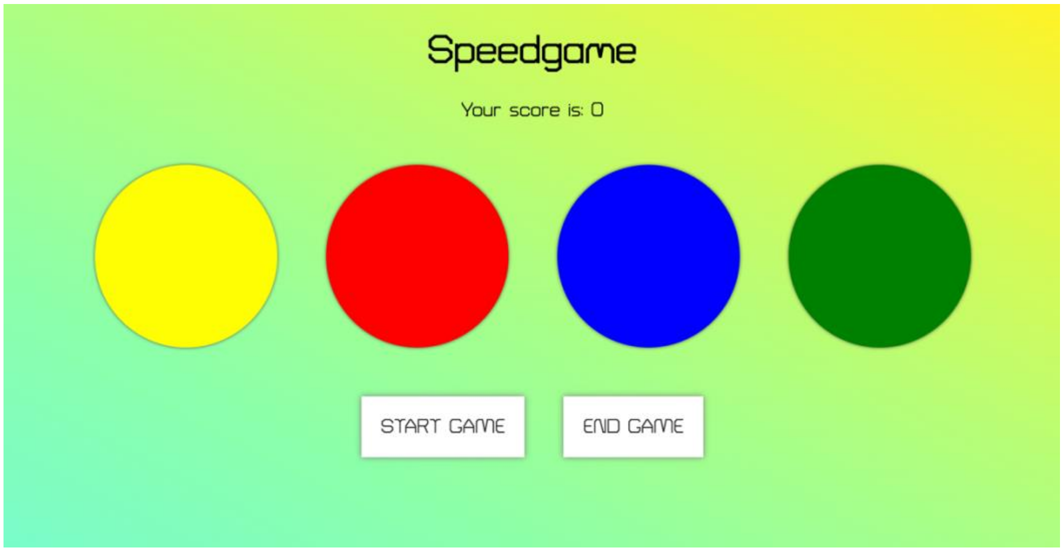
14. **Disable the Start button** during the game

- Use the disabled attribute on the button element, which is checking if the state is true or false.

15. **Disable circle clicks** before the game starts

- Use inline styling

16. Add **styling of your choice**. Here is an example of one possible design:



17. Add sounds to every circle click and an image to for active circle.
18. Add a **screenshot** from your application and add it to the readme file.

Make GitHub commit

This game is a great way to use your knowledge about **function and class components**, **props and states,** and how to **style your application**.

Well done!