Major Challenges:

1. The tasks where I spent more time was trying to find a coordinated and synchronized fashioned way to track and monitor the CTR (click-through-rate) as it requires a certain amount of click events combined with a certain amount of page loads in order to read a user conversion.

2. Determine the way the metrics would be obtained considering that the application serves two different html elements only one would be displayed and both must be tracked separately.

Approach and methods:

Regarding the first challenge, I decided to use local storage as tool of data storage in order to accomplish the next requirements:

- Track of the page-load events: A single function that gets and sets the key value pairs in the local storage combined with a block conditional statement to set the values according to the click-through-rate parameters disfunction would be triggered every time the page is loaded, updating the values when the CTR parameters are yet to be matched and reset to zero when a user conversion reached. When a user conversion is reached, a message will be logged in the console.

- Track of the click events: A second function that keeps track of the clicks while comparing the page loads metric already stored in the local storage. This function is responsible to signal the user conversions when the number of clicks is reached along with the number of page loads. This function is triggered every time the user clicks in the div component.

- Click in Sign-Up button: A specific function is triggered when this is case. Is a direct way of user conversion. This increases the number of converted users x1 automatically and resets the page loads and clicks to zero to start over.

- Random selection of variations: A function will be triggered everytime the page loads in this function we have an array with two strings with the variation names (["control-variation", "test-variation"]), with a *math.random* method we will return randomly a number between 0 and 1 (array length) that represents the index of that array. The value name that corresponds to that index will be stored in a *useState* hook and will be used to set it as a value in the local storage. The function first checks if this item already exists in the local storage, if so, it uses the existing one in order to not change the value.

- Display of "Control-Variation" and "Test-Variation" randomly and separately: Following the previous workflow. As the *useState* hook will be always set to the existing variation defined in the first page load, we use this variable to write a conditional rendering in the JSX.

How to test:

- Different variations or pieces of elements: Once the app is started and opened in the browser you will see one variation displayed, to test and look for the display of the other variation you can open another browser in incognito mode. Since is random, you might have to close and reopen the incognito mode browser a few times until it shows.
- Check the metrics: Opening the Developers tools -> Application -> Storage ->Local storage: You will see the values of *clicks*, *loads*, *convertedUsers*, and variation you are currently seeing. Clicks and loads will update continuously as you refresh the site and click over the DOM once you reach the conditions of a user conversion, these values will be set to Zero and the converted users value will increase +1.

PS. The console also logs the counts but for visual clarity I prefer to see the tracking on the storage panel.

Other Insights:

- This click-through-rate approach could be tested with an E2E testing framework such as playwright.
- I assume Analitycs-api.js would have hosted the functions that handle the fetch/ajax requests in order to connect to a backend environment that stores and processes the metrics obtained.