**Introduction**

This is a simple **beginner-level** project implemented using **React functional components**. In this project, the library **Axios** was used to fetch data from an API and a pagination feature was implemented on the top of the page. The data was an array of 200 elements, each element was a string of ipsum lorem placeholder text. To displayed all the elements on the screen, a component named *List* was built for rendering each element in the array by using the map array method. The *Pagination* component, which was the core of the project, was created for helping user browse through all pages. These two main components were then wrapped parallelly inside a div element in the *App* component.

**Detailed Components Description**

Inside the ***App*** component, the fetched array was stored as a state called *todos* (this name came from the original intention of making this app a todo list) and to prevent infinite loop due to calling the *setTodos* method, the code of fetching and storing data was wrapped inside a useEffect hook which used the URL as the only dependency so it would only run when the URL changed. This *todos* array was shared by *Pagination* and *List*; another two states that were shared by the two components were *pageCapacity* and *currentPageNumber*, which represented, respectively, the number of elements that would be displayed on a single page and where the current page was at.

Inside the ***List*** component, *pageCapacity* and *currentPageNumber* were used to calculated a start index and an end index; these two indices were then used to cut out a portion of the *todos* array with the help of the slice array method; lastly, the cut-out portion would be rendered to the screen in response to user’s selection of current page.

The ***Pagination*** component contained four sub-components. The *Prev* and *Next* components shared a utility component named *PaginationBtn*; these two components were used to help user navigate across the page numbers one by one. The *PageCapacitySelect* component allowed user to select the page capacity; it was also responsible for determining the new current page number after user selected a new page capacity, so it was equipped with a simple logic: if the current page number before the selection of a new page capacity exceeded the last page number after the selection (this could happen because different page capacities come with different page number ranges), then the new current page number would be changed to the new last page number, otherwise, it would stay the same.

The ***challenging*** part of this project lied in the last sub-component of *Pagination* – the *PageNumberBtns* component. This component was responsible for displaying all page numbers as buttons and marking red the page number that was being viewed currently. The challenge was: under the combination of a small page capacity and a large total number of items that needed to be rendered, there could be so many pages that rendering all of these page numbers all at once could result in overwhelming the top section of the web page; therefore, when the amount of page numbers exceeded 10, the length of the range of displayed page numbers would be reduced to 10, so user would only see 10 page numbers at a time. As user navigates across pages between the sixth page and the fourth page from the end, the starting and ending indices of the reduced range would change dynamically in response to the change of the current page number, so that the range would always stay at a length of 10 and put the current page number at the sixth position; this way the current page number would look like it was always in the middle so user could have enough space to move freely and randomly in both directions. This technique was applied by Google at the bottom of their search results page, therefore here I was just trying mimicking what Google was doing because it just seemed like such a popular technique for implementing pagination. The logic of calculating the reduced range was wrapped inside a useEffect hook which took the current page number and page capacity as the dependencies because the range should update according to the change of these two parameters.