**Problem Description:**

1. Create a simple application that fetches data from a public API endpoint and displays it in a list format. However, the API response contains a large dataset, and each item in the list requires heavy computation to display additional details. Your task is to optimize the application using React Native hooks and ensure efficient rendering and data management.
2. On the same screen, at the top, place a counter in the parent component. This counter should increment or decrement upon clicking the + or - buttons, respectively. Apply basic styling to the counter and buttons to make them clearly visible and distinct.
3. When the data fetching begins, a push notification should pop up with the message "FETCHING DATA." After the fetching is complete, another notification should appear stating "FETCHING DATA COMPLETE."

**Purpose** : Ensure that upon increasing or decreasing the counter value, the subcomponents or heavy computations do not re-render.

Requirements:

**Data Fetching and Rendering:**

* Fetch data from the provided API endpoint <https://jsonplaceholder.typicode.com/posts> when the component mounts, using the useEffect hook.
* Display the fetched data in a FlatList component.
* Each item in the list should display the fetched data, including the id and title fields.

**Heavy Computation Optimization:**

* Each item in the list requires heavy computation to display additional details. Optimize the performance by implementing a heavy computation function using the useMemo hook.
* The heavy computation function should accept an item's data as input and return computed details, such as complex calculations or transformations.
* Log the time taken for the heavy computation function to run during each render.

**Callback Memoization:**

* Create a child component that displays details of an item when clicked. Ensure that the child component retrieves data from the API endpoint [https://jsonplaceholder.typicode.com/posts/{postId}](https://jsonplaceholder.typicode.com/posts/%7BpostId) by fetching the post details using the post's ID. Display the same content as shown on the list page for the selected item.
* Pass a callback function from the parent component to the child component using props.
* Memoize the callback function using the useCallback hook to prevent unnecessary re-renders.
* Log a message in the child component when it re-renders due to changes in props.

Ref API Documentation:

JSONPlaceholder API Documentation: JSONPlaceholder API

Set up a basic React Native environment or use an online code playground like Snack by Expo. <https://snack.expo.dev/> and send use the expo link for submission

follow this documentation for pagination

<https://github.com/typicode/json-server/tree/v0?tab=readme-ov-file>

For Example :

1) <https://jsonplaceholder.typicode.com/posts?_page=1&_limit=10> ( Page 1 with limit 10)