**INFINITE SCROLLING**

File 1:

import useBookSearch from "./useBookSearch"

import { useState,useRef,useCallback } from "react"

export const InfiniteScrolling=()=>{

   const [query,setQuery]=useState("")

   const [pageNumber,setPageNumber]=useState(1)

   const{

    books,

    hasMore,

    loading,

    error

}=useBookSearch(query,pageNumber)

   const observer=useRef()

   const lastBookElementRef=useCallback(node=>{

       if(loading) return

       if(observer.current) observer.current.disconnect()

       observer.current=new IntersectionObserver(entries=>{

        if(entries[0].isIntersecting && hasMore){

            setPageNumber(prevPageNumber => prevPageNumber+1)

            // console.log("visible")

        }

       })

       if(node) observer.current.observe(node)

   },[loading,hasMore])

    const handleSearch=(e)=>{

        setQuery(e.target.value)

        setPageNumber(1)

    }

    return(

        <>

        <input type="text" value={query} placeholder="Input your text here" onChange={handleSearch}></input>

        {books.map((book,index)=>{

            if(books.length===index+1){

                return <div ref={lastBookElementRef} key={book}>{book}</div>

            }

            return <div key={book}>{book}</div>

        })}

        <div>{loading && "Loading..."}</div>

        <div>{error && "Error"}</div>

        </>

    )

}

File 2:

import {useEffect,useState} from 'react'

import axios from 'axios'

export default function useBookSearch(query,pageNumber) {

  const [loading,setLoading] = useState(true)

  const [error,setError] = useState(false)

  const [books, setBooks]= useState([])

  const [hasMore,setHasMore] = useState(false)

  useEffect(()=>{

    setBooks([])

  },[query])

   useEffect(()=>{

     setLoading(true)

     setError(false)

     let cancel

     axios({

       method:'GET',

       url:"http://openlibrary.org/search.json",

       params:{q:query,page:pageNumber},

       cancelToken:new axios.CancelToken(c=>cancel=c)

     }).then(res=>{

       setBooks(prevBooks=>{

         return [...new Set([...prevBooks,...res.data.docs.map(b=>b.title)])]

       })

       setHasMore(res.data.docs.length>0)

       setLoading(false)

       console.log(res.data)

     }).catch(e=>{

       if(axios.isCancel(e)) return

       setError(true)

     })

     return()=>cancel()

   },[query,pageNumber])

  return {loading, error, books, hasMore}

}