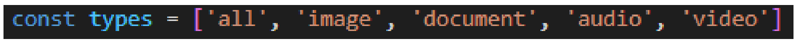
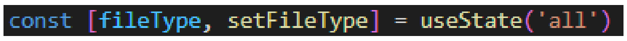
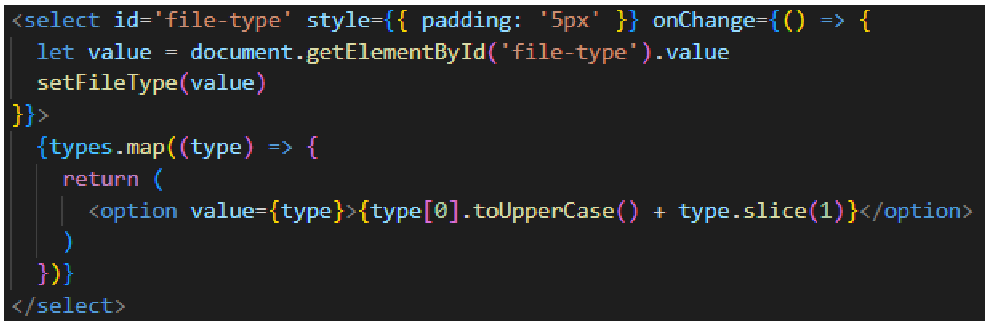
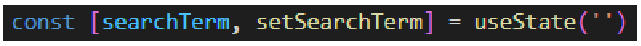
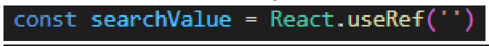
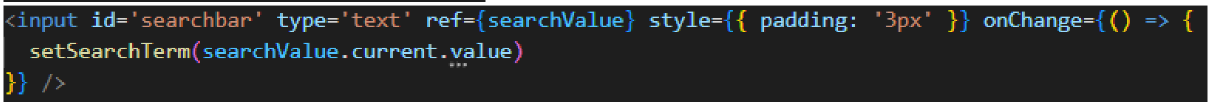
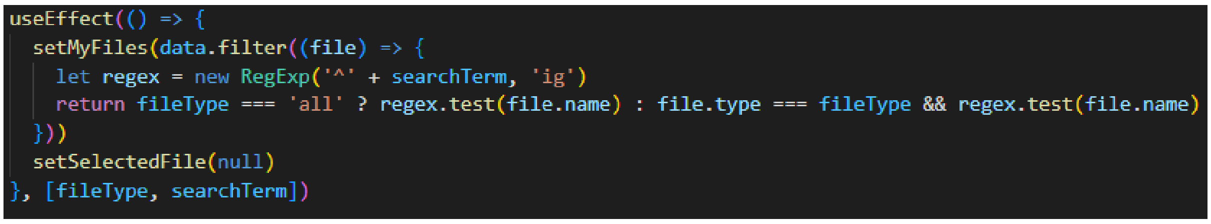
The two features that I have added are filter and search functions. Filter works by showing the files with the file type selected by users while search works by displaying files with the keywords that users enter into the search bar. In my opinion, these two features are quite useful and fundamental in a file manager app and they can come to be very handy when there are many files and users could have a hard time finding the file they want without these 2 features.

For filter functionality:

1. An array of all types of files is created including “all”, “image”, “document”, “audio”, “video”. This array is important as it needs to match with the “file.type”.
   
   ```
   const types = ['all', 'image', 'document', 'audio', 'video']
   ```
2. A “fileType” state is created to store the type that users select with “all” as default state.
   
   ```
   const [fileType, setFileType] = useState('all')
   ```
3. A “select” element with options of all, audio, image, video and document is added next to the delete button. It will set the “fileType” state according to the option chosen by users. The options are coined based on the elements in the “type” array. The code inside the “option” tags is just to capitalize the word.
   
   ```
   <select id='file-type' style={{ padding: '5px' }} onChange={() => {
     let value = document.getElementById('file-type').value
     setFileType(value)
   }}>
     {types.map((type) => {
       return (
         <option value={type}>{type[0].toUpperCase() + type.slice(1)}</option>
       )
     })}
   </select>
   ```

For search functionality:

1. A “searchTerm” state is created to store the type that users select.
   
   ```
   const [searchTerm, setSearchTerm] = useState('')
   ```
2. An “input” element is added next to the select element which will set the “searchTerm” according to the keywords that users enter. The “searchValue” constant is created as a reference to the input element using “React.useRef(“”)” so that the value of the “input” element can be directly referred to when setting the “searchTerm”.
   
   ```
   const searchValue = React.useRef('')
   ```
   
   ```
   <input id='searchbar' type='text' ref={searchValue} style={{ padding: '3px' }} onChange={() => {
     setSearchTerm(searchValue.current.value)
   }} />
   ```
3. Lastly, “useEffect” hook is added which incorporates both filter and search functionalities so that files are shown based on what users enter in the search bar and the file type that users select. It works using a filter method appended to “data”. First, “regex” variable is created based on the “searchTerm” that users enter which is also case-insensitive to test if it matches with the “file.name”. If “fileType” is “all”, then the filter method will only return what is true based on the “regex.test(file.name)” result. Otherwise, it will return based on the “fileType” and “regex.test(file.name)” result. The hook will be prompted to run whenever “fileType” or “searchTerm” changes that is when users type anything in the search bar or select the file type.



```
useEffect(() => {
  setMyFiles(data.filter((file) => {
    let regex = new RegExp('^' + searchTerm, 'ig')
    return fileType === 'all' ? regex.test(file.name) : file.type === fileType && regex.test(file.name)
  }))
  setSelectedFile(null)
}, [fileType, searchTerm])
```