Co-x3 Youtbe API Implementation/ Most Viewed Content Sorting Algorithm

Notes on video_count.tsx:
This file allows you to fetch the Youtbe video view count in respect to the url being pulled from externalContent.json. We are using Youtube's API to pull this data. To do so we need to include specific libraries and pre-install axios in your node.js command prompt.

Imports include:
React Imports: useState and useEffect from the react library.
axios: For making HTTP requests.
content: Pulled from external JSON file externalContent.json.

Main Components Include:
API_KEY: A string placeholder for a YouTube API key.
VideoData: An interface defining the video data structure with url, title, and viewCount.
useEffect executes to fetch video view counts.
fetchViewCounts function: Extracts video IDs from the content array by splitting URLs. If video IDs are available, results in a GET request to the YouTube API using axios.
The **API request** includes parameters to retrieve video title and view count.
Updates videoData state with fetched data if the request is successful (loading/error).

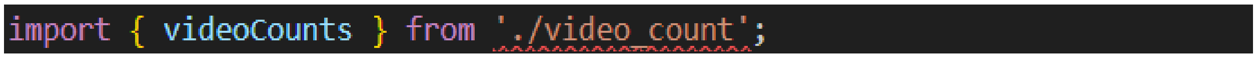
Note: The API_KEY is mentioned but would typically be included in the API request for authentication.

Notes on display_view_main.tsx:
This will output the data in HTML format in which displays the video and its respective number of views. It is pulling the data from the video_count.tsx file which utilizes Youtube's API.

This file is used for validation purposes and to see if the video views are getting fetched and are correctly correlating with the content provided on co-x3's website.

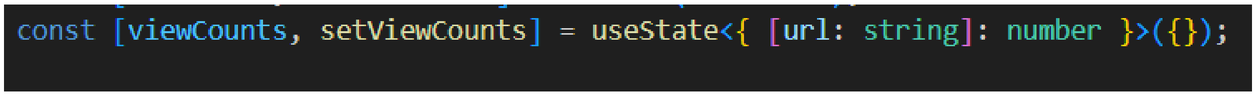
Key changes to useProgramsTable2.tsx now saved as useProgramsTable3.tsx:

Imported videocounts from video_count.tsx file **(line 3)**:



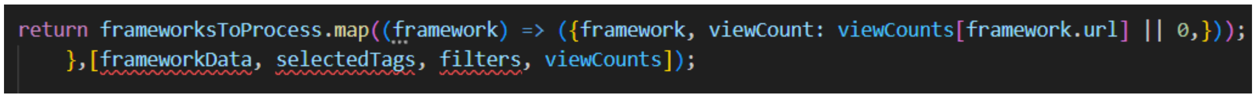
```
import { videoCounts } from './video_count';
```

Fetching views in respect to content urls from the externalContent.json file **(line 32)**:



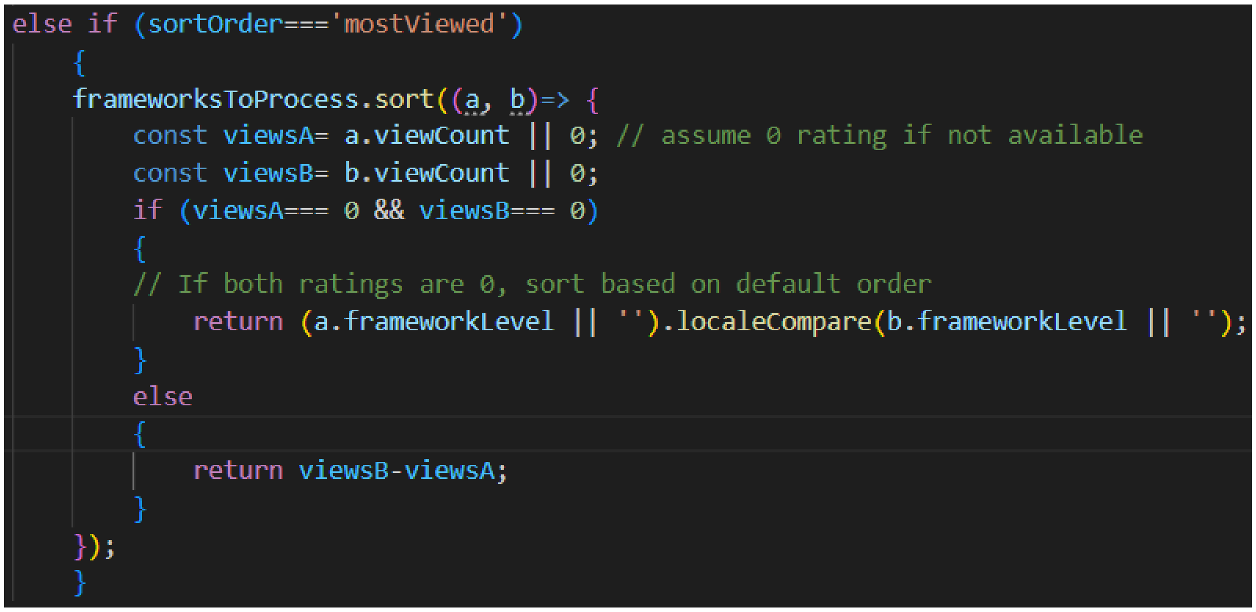
```
const [viewCounts, setViewCounts] = useState<{ [url: string]: number }>({});
```

Changing framework to account for pulling view count in respect to content urls from the externalContent.json file **(line 84-85)**:



```
return frameworksToProcess.map((framework) => ({framework, viewCount: viewCounts[framework.url] || 0,}));
    },[frameworkData, selectedTags, filters, viewCounts]);
```

Sorting Algorithm when the mostViewed option is selected. If there are no views, content will be sorted based on default sort. Else it will sort by highest views to lowest views **(lines 145-160)**.



```
else if (sortOrder==='mostViewed')
    {
    frameworksToProcess.sort((a, b)=> {
        const viewsA= a.viewCount || 0; // assume 0 rating if not available
        const viewsB= b.viewCount || 0;
        if (viewsA=== 0 && viewsB=== 0)
        {
        // If both ratings are 0, sort based on default order
            return (a.frameworkLevel || '').localeCompare(b.frameworkLevel || '');
        }
        else
        {
            return viewsB-viewsA;
        }
    });
    }
```