

# Requirements

**Create a typescript Vue/Typescript application to fetch the top 25 posts in a reddit community and display the JSON response**

- ✓ a. Use this endpoint to get the 25 posts for a community:

[https://www.reddit.com/user/{community\\_name}.json](https://www.reddit.com/user/{community_name}.json)

i. E.g. the computerscience subreddit page can be accessed as JSON like this:-

<https://www.reddit.com/r/computerscience.json>

✕ Headers Payload Preview Response Initiator Timing Cookies	
▼ General	
Request URL:	https://www.reddit.com/r/computerscience.json?limit=25
Request Method:	GET
Status Code:	● 200 OK
Remote Address:	146.75.81.140:443
Referrer Policy:	strict-origin-when-cross-origin
▼ Response Headers	

- ✓ b. Use either Axios or the Fetch API to make HTTP requests to the JSONPlaceholder API endpoints.

```

import axios, { AxiosInstance } from 'axios'
import ResponseDefault from '../types/ResponseDefault'

You, 2 days ago | 1 author (You)
export default class BaseService {
  public http: AxiosInstance

  constructor() {
    this.http = axios.create({
      baseURL: import.meta.env.VITE_APP_API_URL,
    })

    //We can add some interceptor code h
  }

  public apiErrorTreatment(exception: any): ResponseDefault {
    if (exception?.response?.data) {
      return exception.response.data as ResponseDefault
    } else {
      return new ResponseDefault(exception.response?.status, exception.message, exception.payload)
    }
  }
}

```

- ✓ e. Utilize Vue.js lifecycle hooks such as created or mounted to fetch data when the component is created or mounted.

```

mounted() {
  if (!this.user) {
    this.$store.dispatch(ActionTypes.GET_USER_BY_NAME, this.userName);
  }
},

```

## Suggested Features

- ✓ a. Create typescript interfaces for the data being fetched

```

export interface Post {
  kind: string,
  data: { ...
}

}

```

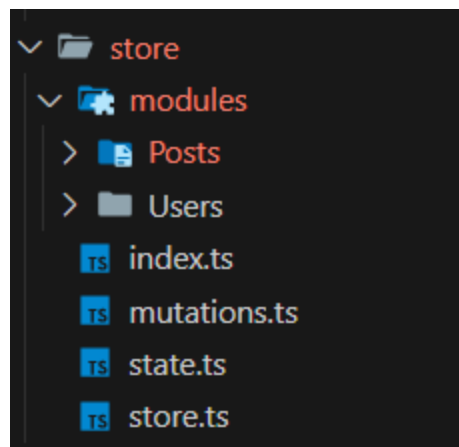
You, 2 days ago | 1 author (You)

```

export interface ApiResponse {
  kind: string,
  data: {
    after: string,
    dist: number,
    modhash: string,
    geo_filter: string,
    children: Array<Post>,
    before: string
  }
}

```

- ✓ b. Usage of a frontend store is encouraged (Pinia or Vuex preferred)



- ✓ c. Usage of a frontend framework is encouraged (Vuetify preferred; Bootstrap-vue, Primevue, etc)

```
const vuetify = createVuetify({
  components,
  directives,
  icons: {
    defaultSet: 'fa',
    aliases,
    sets: {
      mdi,
      fa
    }
  },
})
```

- ✓ d. Add a search input field that filters down the results on the frontend



**mobotsar** Jan 15, 2023

**I looking for books, videos, or other res**

- ✓ e. Allow users to sort information

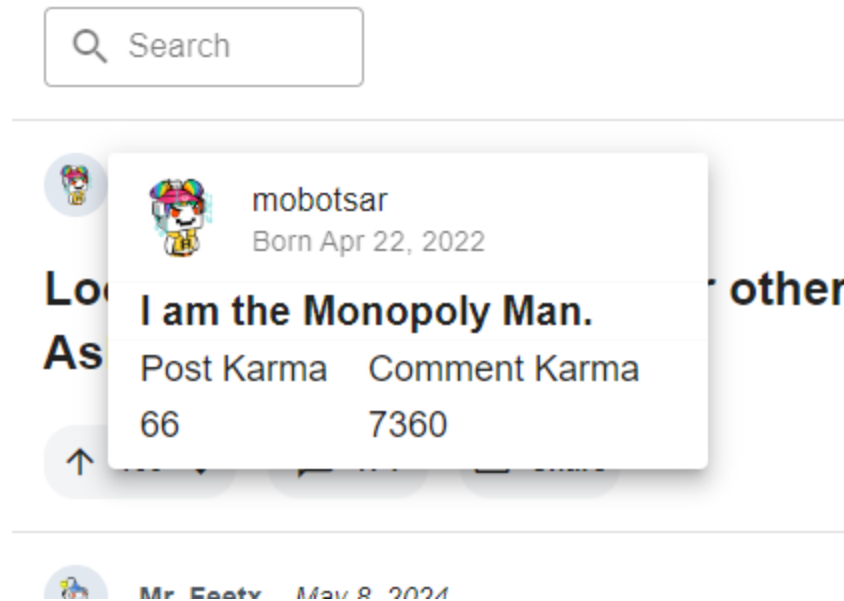
Date ↑

...

**veral topics?**

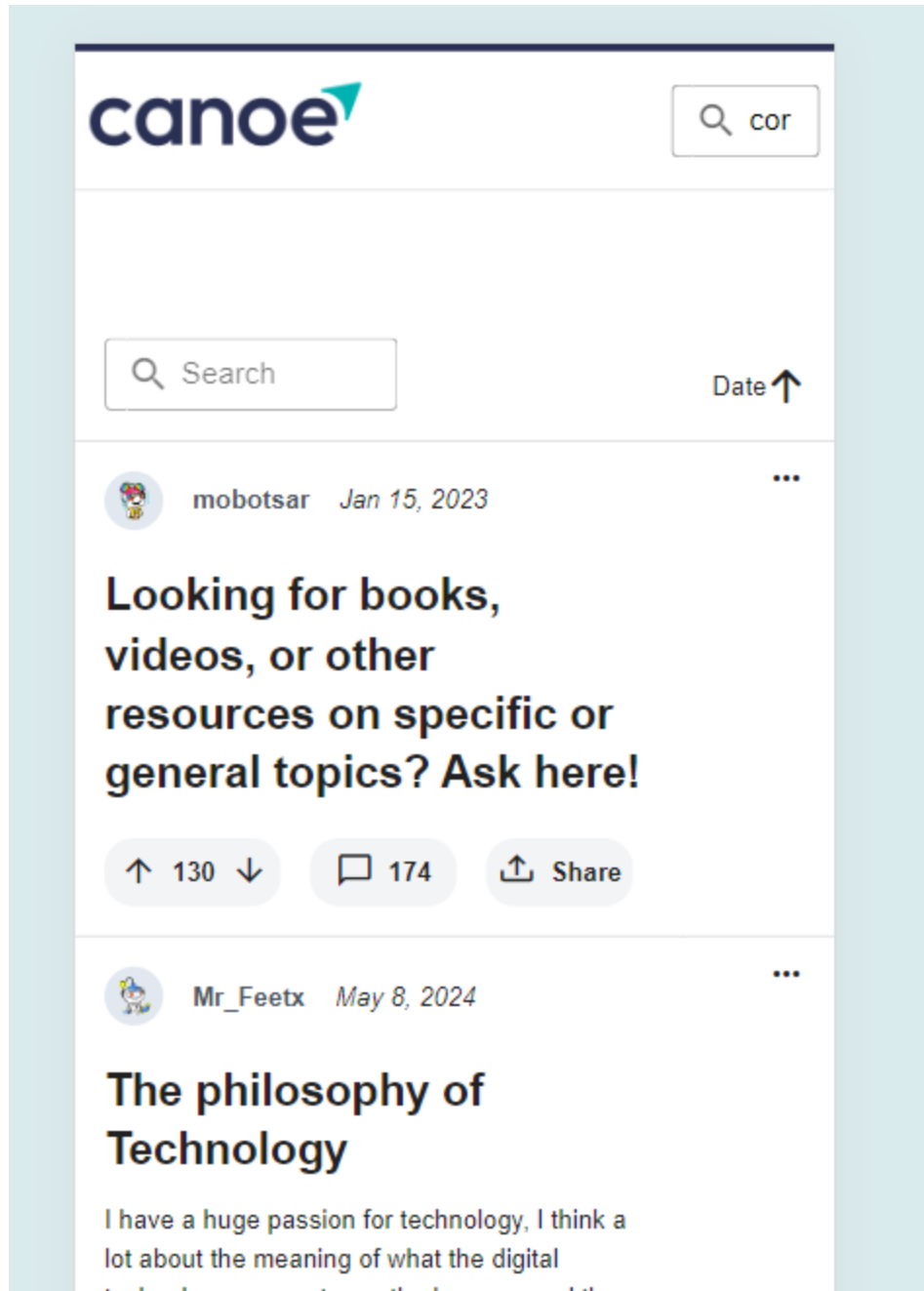
- ✓ f. Drill downs: When showing posts in a community, let the user click into the author (either as a new page or a modal). Use this endpoint to drill into the post author:-

<https://www.reddit.com/user/{username}.json>




## Bonus features (not required)

- ✓ a. Responsive design



☒ b. Error handling

 Community not found.  
Please, try to enter other community name.

☐ c. Row selection

☒ d. ~~Pagination controls~~

 Lifehater007 May 8, 2024

...

**Why are algorithms called algorithms? A brief history of the Persian polymath**

↑ 16 ↓

🗨 1

🔗 Share

< 1 2 3 4 5 6 >

☐ e. Data Export

## What We Are Looking For

☒ ~~Clear instructions for running and testing your solution~~

☒ ~~Insight into your thought process for the design choices and tradeoffs you make~~

☒ ~~Cleanliness of code and adherence to standard principles such as DRY/KISS/YAGNI/SOLID/etc...~~

☒ ~~Organization of your code into reusable components for fetching and displaying data~~

☒ Separation of concerns

- ✓ ~~Readability of code~~
- ✓ ~~Testability of code~~
- ✓ ~~Approach to error handling~~

## Scalability considerations:

### **How will your application work as the data set grows increasingly larger?**

The application is expected to perform adequately with growing data sets. However, it would be beneficial to implement a more robust pagination strategy in alignment with backend capabilities. Additionally, integrating server-side caching would enhance performance by reducing redundant queries. In my particular case, this was very useful since I always request user data from posts that are visible. If I would request all users from all posts, it would take much more time and not be performative at all.

### **How will your application work as the number of concurrent users grows increasingly larger?**

To handle a larger number of concurrent users effectively, implementing queues, possibly utilizing Redis, would be necessary. For frontend scalability, employing web workers could optimize performance by offloading heavy computations from the main thread. In my particular case, this was very useful since I always request user data from posts that are visible. If I would request all users from all posts, it would take much more time and not be performative at all.

### **If you implement any of the bonus features, please mention them in your readme with a brief explanation of your design choices.**

In the bonus section, I have implemented responsive design, error handling, and pagination.

Responsive design is fundamental as it ensures accessibility across various devices, enhancing reliability and user experience. I agree with the mobile-first methodology, so I always consider implementing a responsive layout to make the application accessible for all users, thereby increasing reliability.

Error handling is crucial for providing users with feedback even in failure scenarios. By employing catchers and error treatment functions, we mitigate

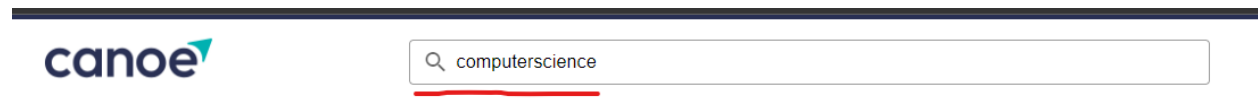


crashes and ensure a smoother user experience. It's essential to address as many error cases as possible to maintain application stability.

Pagination significantly improves performance by limiting data retrieval to what is immediately needed. In my case, it optimizes performance by fetching only the data visible in the current view, rather than loading all data at once, which would be less efficient. In my particular case, this was very useful since I always request user data from posts that are visible. If I would request all users from all posts, it would take much more time and not be performative at all.

## Extra feature!

I'd like to mention that I have added a additional functionality. Check below



we can change subreddit subject.