COMP 4350 – Assignment 1

Zack Holmberg
7777823

GitHub Repository Link: https://github.com/ZackHolmberg/comp4350a1

DockerHub Link: https://hub.docker.com/r/zackholmb/comp4350a1

Prerequisites

Must have Docker installed. If you don't, get it here.

Running the Container

To download and run the docker image containing my implementation, simply copy, paste and execute the following command in your terminal:

```
docker run -p 8080:8080 zackholmb/comp4350a1
```

After the image has been downloaded and started, go to

http://127.0.0.1:8080

in your web browser. The web application will be running.

How to use the web app

To use the web application, enter the tag you want to search for in the input field at the top of the page. Once you have entered your tag, click the button to the right of the input field. This will trigger a search for posts relevant to the tag. The app will also start tracking the response time, which will be displayed at the bottom of the page.

Once the search completes, elements will be generated on the page that you can interact with. For each post that is retrieved from the search, a collapsible element will be generated. This element will contain the title of the post, as well as its creation date and score. To view the content of the post, hover over the post you'd like to view and click it. The collapsible will expand, and the question's body, comments, answers and answers' questions will be displayed.

Implementation

Frameworks

For my implementation I went used `Vue.js` to build a web application. Additionally, the web app is hosted using `Node.js`. Finally, the app uses `Axios` to make API calls.

Logic

The logic under the hood of my Web App is quite simple and straight forward. When a user enters a tag into the input field and clicks the button to execute a search, the app gets the data from the input field and uses it in two calls that are made to the Stack Overflow API.

The first call gets the most recently created questions in the last week. To calculate the date that is used in the call, the app calculates the epoch value representing a one week prior to the current date. Once the first API call returns, the app adds the list of data to a temporary local variable. Next, the app executes the second API call, which retrieves the question posts with the highest scores in the last week. Then it adds this list to a local variable. The app also tracks how long it took to retrieve data from the two API calls and process it, which it displays on the web page at the bottom.

Once the app has the data from the two API calls, it takes the top 10 items from each list (removing duplicates) and adds them to a list containing the items to be displayed. Finally, the toDisplay list is sorted by creation date in descending order, and each post's creation date is converted from the epoch format to a human-readable format.

Once the toDisplay list has finished all of the processing described above, the rendering logic begins. The toDisplay list is taken and Vue shorthand is used to create a loop that generates a `Collapsible` component, passing the list object (which represents a question post) to the component.

The `Collapsible` component takes the object that is passed into it and creates elements to display all of its data, including the posts' title, score, creation date, and body content. Additionally, the component creates elements for the title, score, creation date, and body content for all of a posts' comments, answers and answers' comments.