Challenge: Commenting Feature

This lesson contains challenges that you can keep on implementing in the application to enhance your skills.

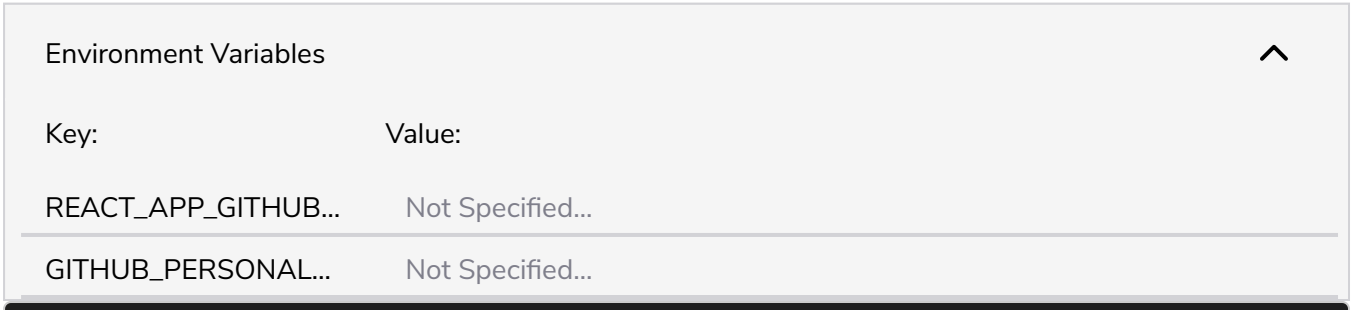
This last lesson is for hands-on experience with the application and implementing features yourself. I encourage you to continue implementing features for the application and improving it. There are a couple of guiding points to help you with implementing the Commenting feature. In the end, it should be possible to show a list of paginated comments per issue on demand. Finally, a user should be able to leave a comment. The source code of the implemented feature can be found here.

- Introduce components for fetching a list of comments (e.g. `Comments`), rendering a list of comments (e.g. `CommentList`), and rendering a single comment (e.g. `CommentItem`). They can render sample data for now.
- Use the top level comments component (e.g. Comments), which will be your container component that is responsible to query the list of comments, in the *src/Issue/IssueItem/index.js* file. In addition, add a toggle to either show or hide comments. The IssueItem component has to become a class component or needs to make use of the `withState` HOC from the recompose library.
- Use the `Query` component from React Apollo in your container Comments component to fetch a list of comments. It should be similar to the query that fetches the list of issues. You only need to identify the issue for which the comments should be fetched.
- Handle all edge cases in the Comments to show loading indicator, no data, or error messages. Render the list of comments in the `CommentList` component and a single comment in the `CommentItem` component.
- Implement the pagination feature for comments. Add the necessary fields in the query, the additional props, and variables to the `Query` component, and the reusable `FetchMore` component. Handle the merging of the state

in the `updateQuery` prop.

- Enable prefetching of the comments when hovering the “Show/Hide Comments” button.
- Implement an AddComment component that shows a `text-area` and a `submit` button to enable user comments. Use the `addComment` mutation from GitHub’s GraphQL API and the Mutation component from React Apollo to execute the mutation with the submit button.
- Improve the `AddComment` component with the optimistic UI feature (perhaps read again the Apollo documentation about the optimistic UI with a list of items). A comment should show up in the list of comments, even if the request is pending.

Here you go:




```
import React from 'react';

import Link from '../../Link';

import './style.css';

const Footer = () => (
  <div className="Footer">
    <div>
      <small>
        <span className="Footer-text">Built by</span>{' '}
        <Link
          className="Footer-link"
          href="https://www.robinwieruch.de"
        >
          Robin Wieruch
        </Link>{' '}
        <span className="Footer-text">with &hearts;</span>
      </small>
    </div>
    <div>
      <small>
        <span className="Footer-text">
          Interested in GraphQL, Apollo and React?
        </span>{' '}
        <Link
          className="Footer-link"
```

```
          href="https://www.getrevue.co/profile/rwieruch"
        >
          Get updates
        </Link>{' '}
        <span className="Footer-text">
          about upcoming articles, books &
        </span>{' '}
        <Link className="Footer-link" href="https://roadtoreact.com">
          courses
        </Link>
        <span className="Footer-text">.</span>
      </small>
    </div>
  </div>
);

export default Footer;
```

I hope this lesson, building your own feature in the application with all the learned tools and techniques, matched your skills and challenged you to implement React applications with Apollo and GraphQL.

I would recommend working to improve and extend the existing application.

If you haven't implemented a GraphQL server yet, find other third-party APIs that offer a GraphQL API and build your own React with Apollo application by consuming it. Keep yourself challenged to grow your skills as a developer.