Today, I mainly focused on implementing the search feature and tidying up the Edit a Post feature interaction.

Regarding the search feature, I break it down into four main tasks.

First, I created two search APIs: one is searching Posts, and the other one is searching Users. I used the fuzzy match to retrieve the value in the associated property in the Posts and Users collections. Then, I used the Postman to test the APIs. It can retrieve data successfully.

Second, I created a search context that provides search results and search functionality. Regarding search functionality, I used the `axios.all` to request two APIs parallelly. Then save the response data in this format: `{ posts: postsData, users: usersData }`. The search context provider allows the search page and navbar to consume the `search result` and `search functionality` values using the `useContext(SearchContext)` hook.

Third, I updated the Navbar with search functionality. I used onChange to monitor the value change in the input component and useState to save the value as the query. I got the search functionality through the `useContext(SearchContext)` hook. Then, in the handle search function, I used the search functionality to request and useNavigate to navigate to the search result page with the query as the params. Then, set the query to null.

In the final step, I got the search results through the `useContext(SearchContext) hook` and used the basic map method to render the data on the search result page.

For the Edit a Post feature, I stuck at the style part. I used an MUI Popover component to render a pop-up. Then, the style of the other component in the Popover component did not work successfully. I spent several hours writing a native pop-up from scratch to keep the style working. Fortunately, I solved this issue and improved the user interface.

Overall, I enjoyed the coding time today and will refine the search page interface and interaction tomorrow.