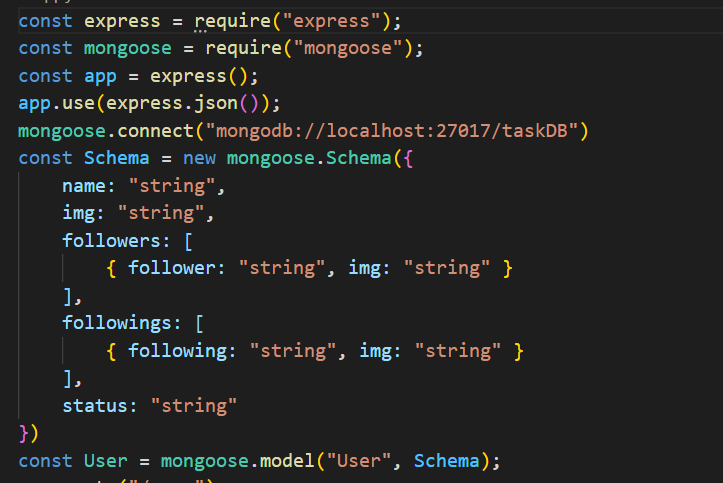
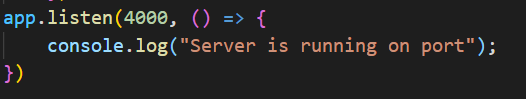
**SERVER**

* This is the basic setup of the backend where I install and require the **npm packages**, created a **schema**, **module** to describe the format for storing the details in the database, **connected** with **MongoDB** local server and listening to the port server at 4000.

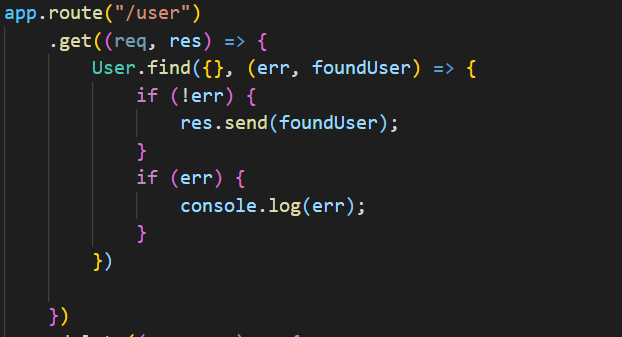
It has 5 objects:

1. “name”: It will store the name of the users.
2. “img”: It will store the image URL of the same
3. “followers”: It is an array of followers of the respective user which contains their name and image URL.
4. “followings”: It is an array of followings of the respective user which contain their name and image URL.ss





* Then I created the route: **“/user**” where I ask to **get** requests for all the accounts of users present in the database using find syntax with no-parameter and pass it to the frontend using **react. send .**

****

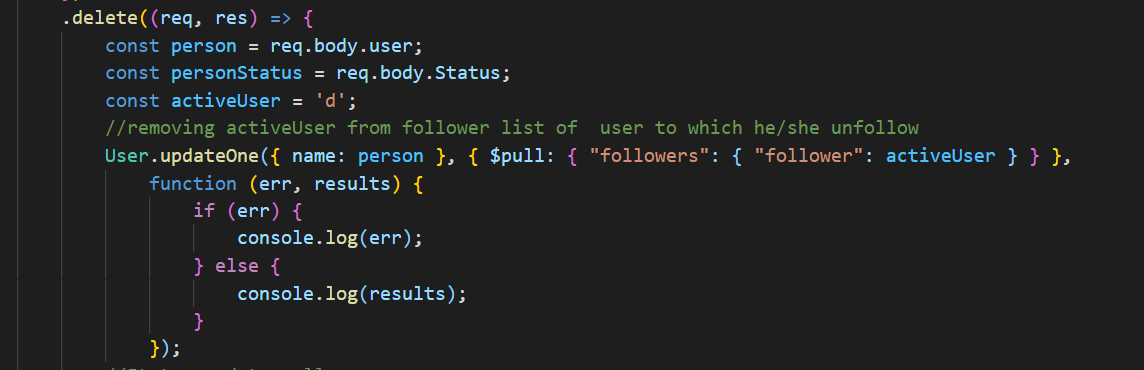
* Now as to add and delete user running the app in the followers’ list of other user accounts.

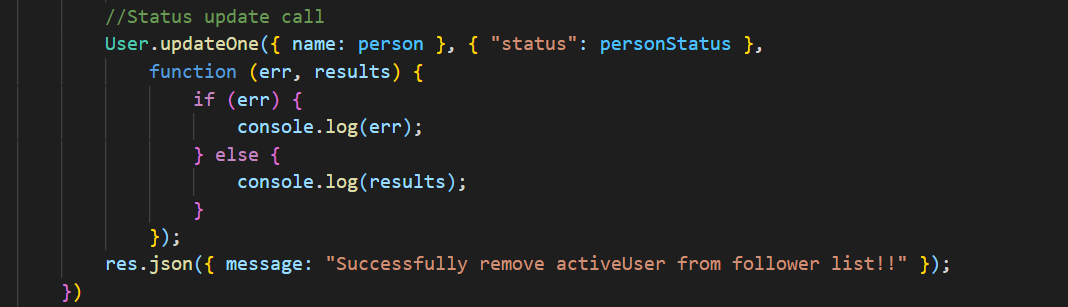
I ask for 2 request: delete, patch. Here, I have taken user ‘d’ who is performing the task.

1. In **patch:** I used **updateOne** operation where it will search for the account that user followed and add him/her in the followers’ list of that particular account user using **$push** operation and updated the status from follow to following.

****

1. In **delete**:I used **updateOne** operation where it will search for the account that user followed and delete him/her from the followers’ list of that particular account user using **$pull** operation. After that, I update his/her status from following to follow.

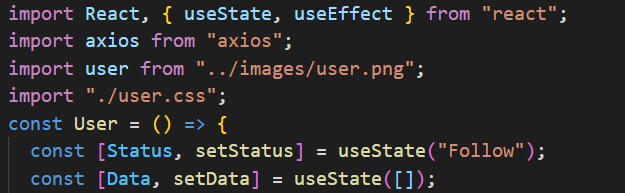
****

****

**CLIENT**

* Starting with frontend in react I imported some packages in **jsx** file and created 2 hooks using **useState** in react:

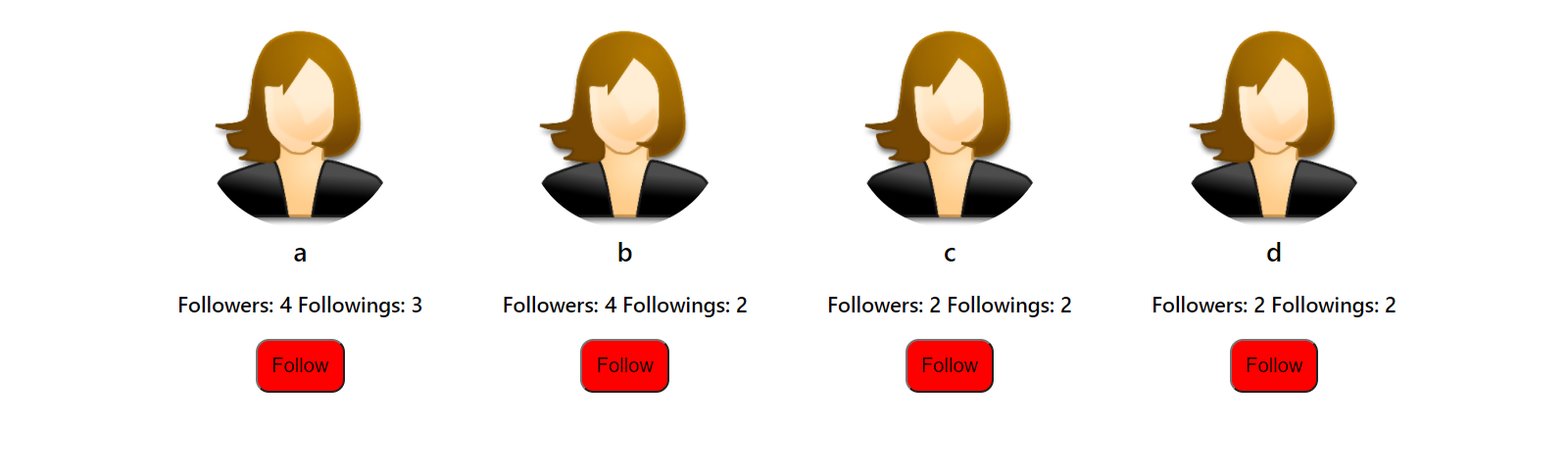
1. “status”: It will update the status of particular account from follow to following and vice versa.
2. “Data”: It will store data coming from the backend so as to display the same on the page to the client-side.



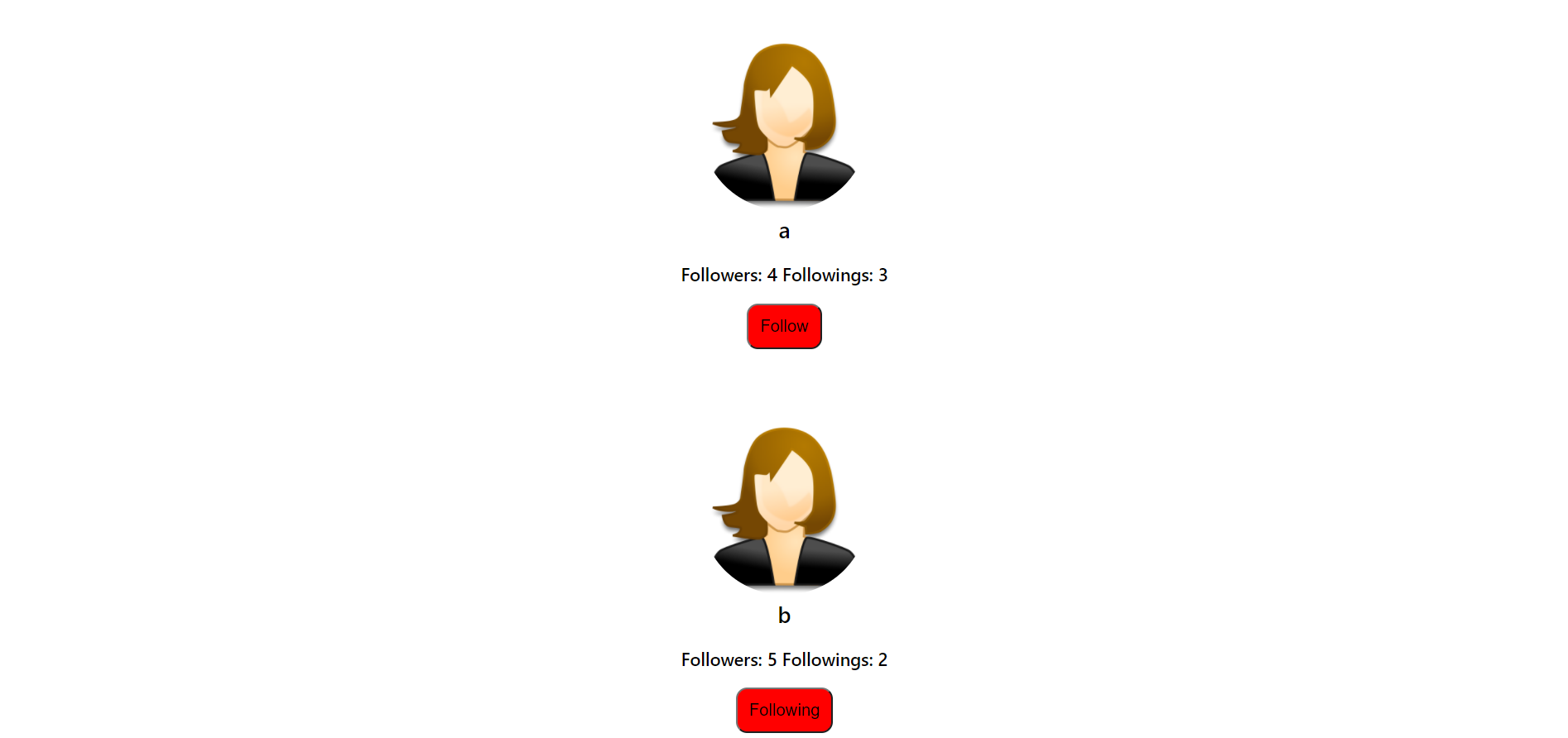
* This is the basic HTML code with some **react mapping** function so as to avoiding writing the same piece of code multiple times.

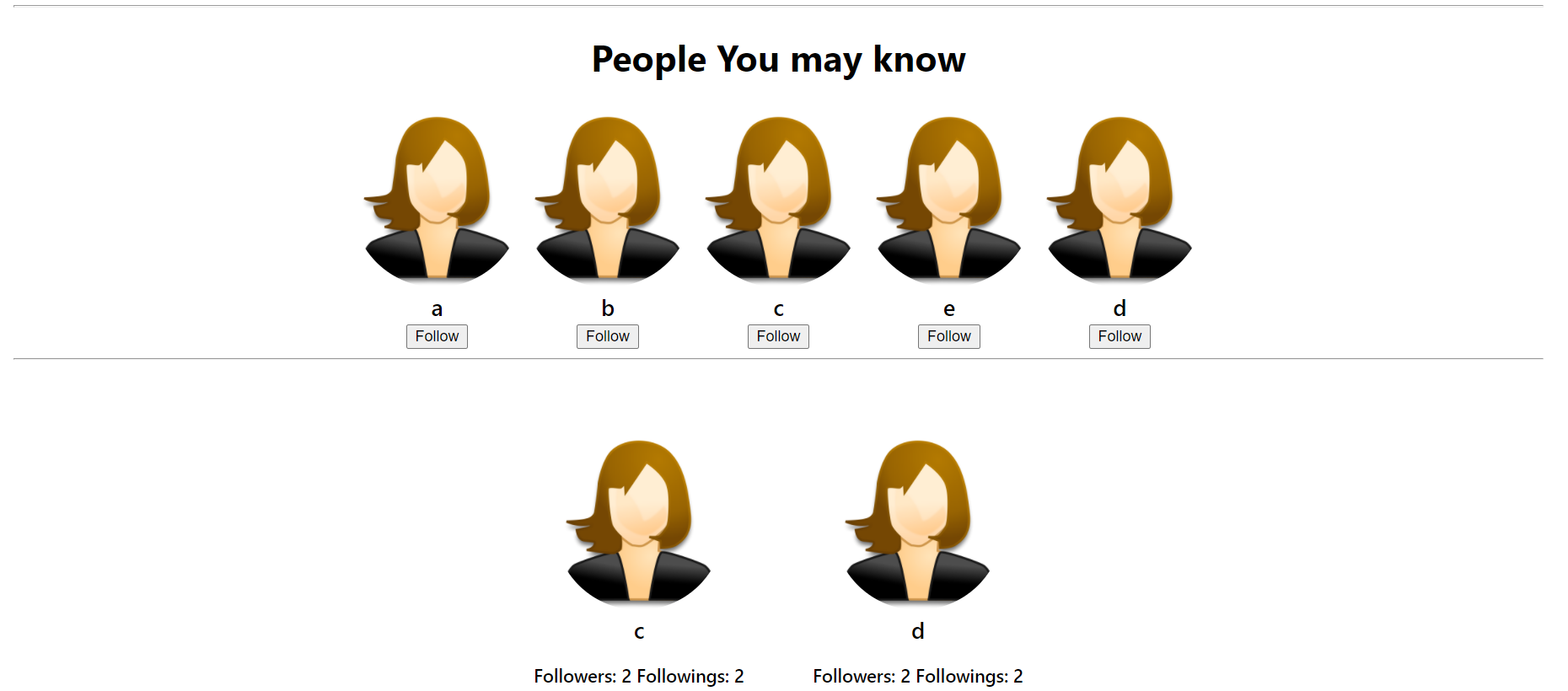
1. In this it will display all the accounts to the user to which he/she can follow.
2. If the user follows the particular account then he will be able to **see the follower list** of that **particular account** as a suggestion (he/she might know) and he/she will be **added to the follower list** of the account and thus resulting in incrementing the length of the array of followers of that account. Therefore, **followers number will increase to 1.**
3. If the user unfollows it then again, he will not be able to see the follower list of that account which leads to the **removal of his/her name from the follower list** and thus results in a **decrement of followers number by 1.**

**Initial display to the user:**

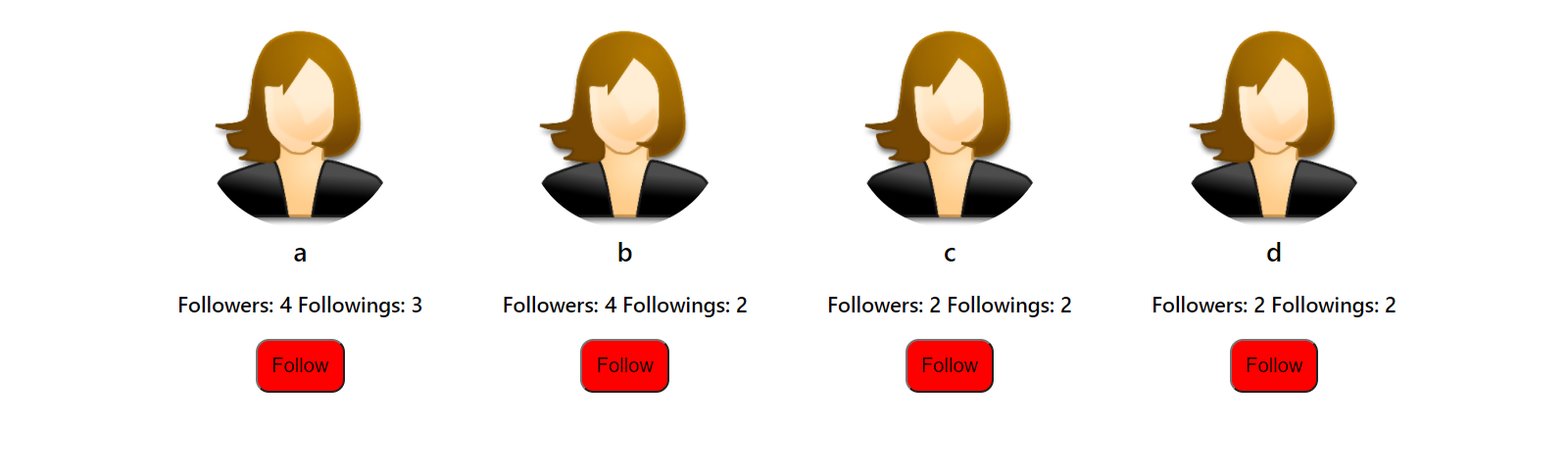


**After he/she follows:**





**After he/she unfollows back:**



User

Followers: number Followings: number

Button(Follow/Following)

Following Not Following

Suggestions

(Follower list)

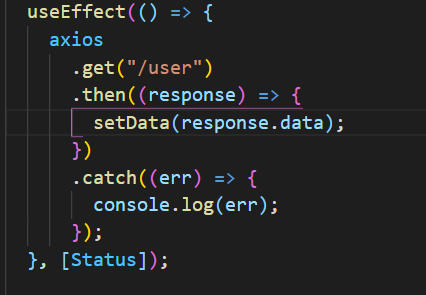
Account1 Account2 and so on

Will not display suggestion

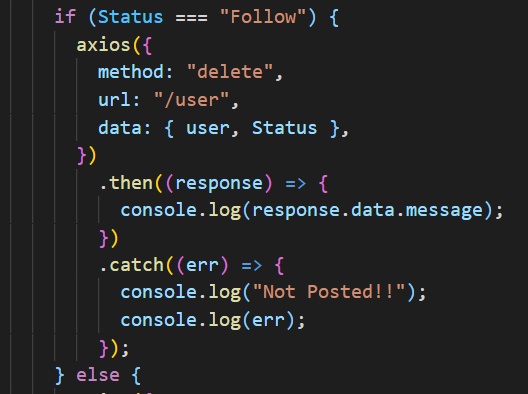


* I have used axios to fetch the backend route data to frontend.

1. This is the get request using axios in the useEffect which will render as the status will update and set the Data hook to the user data coming from backend to frontend.



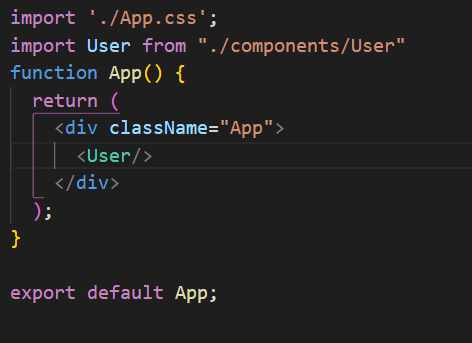
1. This is the delete request which will request if status===”Follow” so as to delete username from that account followers’ list.



1. This is the patch request which will request if status===”Following” so as to add username to that account followers’ list.



* Rendering all the data from User.jsx file to App.js.



**\*\*\*Suggestion is given on the basis of the follower list of the account that is followed\*\*\***