# Frontend

## Key points:

* We use {post?.desc} in order to extract the description of the post from the dummy data. Here, ‘?’ indicates that the description can be empty.
* Using profile as prop in rightbar enables us to show the content of the right bar accordingly.
* The profile page has a different right bar and the home page has a different right bar. So, once a user clicks on the profile the contents inside the rightbar change dynamically.
* We use the assets folder location as .env file to ensure that every asset used is fetched accurately.

## Connecting api with react app

* For making requests, we use axios and not JS fetch.
* We add,

"proxy": "http://localhost:8800/api"

In the package.json file of the client so that the client and server sides can run independently during development. We could run them together but for that we would need to configure CORS.

* If we are using something changeable inside the useEffect hook, we need to add that variable as dependency so that it can be re-rendered again and again.

## Retrieving the posts according to the user logged in

* We use context api in order to retrieve the userId from the user.
* What is **context state**? When we login the system, then our user data is stored in Redux/context state which then can be used anywhere in the application.
* Context state acts like a common state for the whole application due to which we need not use props for passing.
* Here in the routes of the user in our server side, we set query:

const userId = req.query.userId;

    const username = req.query.username;

* This query allows us to retrieve not only the userId of the user but also the username of the user that is logged in.

## Context api

* In the AuthContext.js file, we have declared a provider. Here the provider acts like a wrapper and the values written inside the provider is available to all the children inside it.
* For example, we wrap index.js file with AuthContextProvider, then the values like user, isFetching, error, dispatch can be used by the whole application anywhere. So, we will not need to pass certain properties as props one by one.
* Whenever a user login, actions are dispatched.
* Then the actions are read by the reducers and we update the properties inside the INITIAL\_STATE. Reducers updates the INITIAL\_STATE into a NEW\_STATE.
* If the user exists then the user updates the user.state into the user actually login inside the application and if there’s failure then, error becomes true and isFetching becomes false.
* After creating the context and actions, we go to App.js where we use the context from AuthContext which is user.
* So, if the user is present (i.e., logged in) then we redirect the user to home page through ***Navigate***in react-router-dom or else we send it to register page which means the person needs to create an account first.