## Setup Instructions

1. Install NVM: <https://github.com/nvm-sh/nvm>
2. Ensure you’re running Node v10.15.3+ and npm v6.4.1+.

## Assignment

1. Create a React Class for a Social Card for a GitHub User.

<https://developer.github.com/v3/users/#get-a-single-user>

It should display the following attributes for a user:

* Avatar
* Username
* Location
* Following/Followers
* Bio
* Link to the Github Page
* Blog
* Email

1. Add the `react-router` package for routing management and add a Navigation menu to your app.
2. Create a GitHub Profile Viewer
   1. Users can search for a GitHub profile - when you put in a username into the search field it displays their profile on the page.
   2. User can see the error if the profile doesn't exist.
   3. User can see the information of the searched person - Use React Router to navigate directly to a given page to view Any User via their username. E.g.: [http://localhost:3000/<username>](http://localhost:3000/%3Cusername%3E)
3. Add authentication to your app.

For the scope of this assignment, you don’t need to implement GitHub OAuth. You can authenticate a user using their personal access tokens: <https://developer.github.com/v3/auth/#via-oauth-and-personal-access-tokens>

1. Create a login page where the user can enter their username and password (Personal access token of the user).
2. Validate the username using the same get user API and store the access token for further requests.
3. Create a logged-in route to display the logged-in user’s profile.
4. Implementing a "*Who to follow*" suggestions box (using react hooks) <https://developer.github.com/v3/users/#get-all-users>

Note: To keep it simple, you don’t need to add the checks if the user is already following another user.

* 1. On startup, load accounts data from the API and display any 3 suggestions.
  2. On clicking "Refresh", load 3 other account suggestions into the 3 rows
  3. On click 'x' button on an account row, clear only that current account and display another
  4. Each row displays the account's avatar and links to their page

**Note**: UI would be similar to twitter’s ‘who to follow’ suggestion page.



1. Test Cases for each component.

List of pages to be created

1. User profile page for an arbitrary user with a search bar for searching a user.

2. Login page.

3. A page to show currently logged in user's profile. (logged in route)

4. A page to show suggestions of who to follow. (logged in route)

### Good to have

1. Allow a user to actually follow another GitHub user using the API.
2. List followers of a user.
3. Autocomplete for the search bar.

## Resources

### JS

[JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript)

[The Modern JavaScript Tutorial](https://javascript.info/)

[W3Schools JavaScript Tutorial](https://www.w3schools.com/js/)

[ES6 Syntax and Feature Overview](https://www.taniarascia.com/es6-syntax-and-feature-overview/)

### React and Redux

Official Docs: [Hello World](https://reactjs.org/docs/hello-world.html)

API ref: [React.Component](https://reactjs.org/docs/react-component.html)

A good read: [Thinking in React](https://reactjs.org/docs/thinking-in-react.html)

[Tutorial: Intro to React](https://facebook.github.io/react/tutorial/tutorial.html)

[Redux - A predictable state container for JavaScript apps.](http://redux.js.org/)

[Redux - Core Concepts](https://redux.js.org/introduction/core-concepts)

## Concepts to be tested

* API integration.
* Proper routing.
* Test Cases.

### Components

* It is pure, meaning that all information required to render is passed in by props.
* Has no direct store interaction. Use a container to wrap the component if needed.
* Has component tests.
* It is generic and re-usable when possible.
* Has documented props. (in PropTypes)

### Using a Container

* It needs some data injected into its props that the parent component doesn’t have access to.
* Needs to be able to perform some sort of action that affects the state of the store.

### Implementing a component

* All data for rendering must come from props.
* Local rendering information that does not affect the store may be kept in state.

### React

* Initializing state in React
* Using arrow functions to bind class methods
* How functional setState works, as well as plain setState
* Handling changes
  + Component Lifecycle methods
  + Functional components with Hooks API

### Redux

* State for the whole application is stored via Redux Store
* Avoid Mutating the Store
* Emit Actions to update the state
* Write pure Reducers to transform the state
* Using Selectors to extract data from state