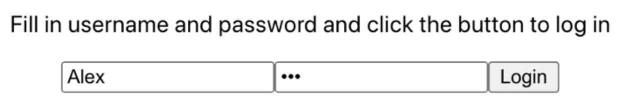
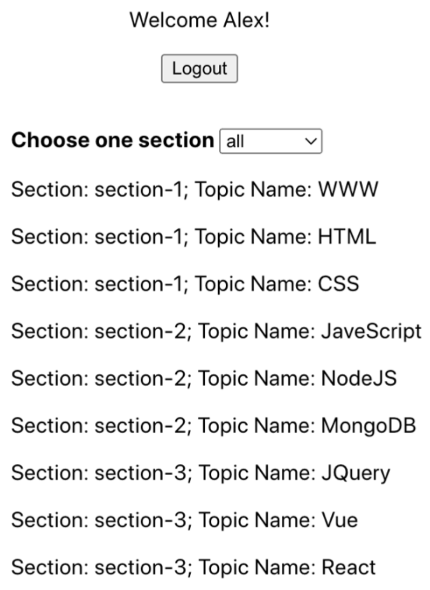
COMP3322A Modern Technologies on World Wide Web
Lab 7: React

Overview
In this lab exercise, we will use **React** to implement an interactive web page, which displays a list of course topics. We first login with a valid username and corresponding password, and then we will see the topic list and can filter these topics by choosing a section name from a drop-down list. Please refer to the following screenshots:


Fill in username and password and click the button to log in

Alex ••• Login

Upon initial page load, you will see a login page with two input boxes and a login button. You can fill in the correct username with its password and click the login button to log in.


Welcome Alex!

Logout

Choose one section all

Section: section-1; Topic Name: WWW

Section: section-1; Topic Name: HTML

Section: section-1; Topic Name: CSS

Section: section-2; Topic Name: JaveScript

Section: section-2; Topic Name: NodeJS

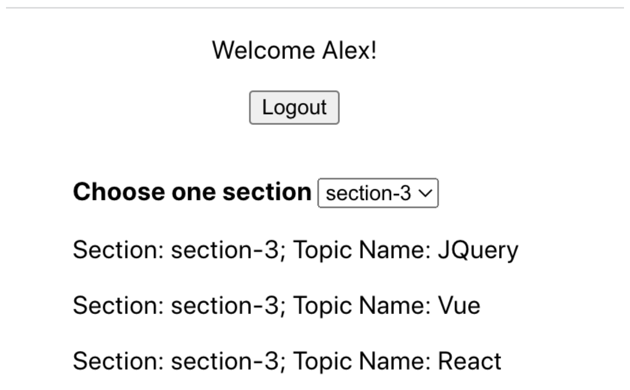
Section: section-2; Topic Name: MongoDB

Section: section-3; Topic Name: JQuery

Section: section-3; Topic Name: Vue

Section: section-3; Topic Name: React

After successfully logged in, you will see a welcome message, a logout button, a drop-down list with the default option “all”, and a list of topics right below showing the section that the topic belongs to (chosen from three sections: section-1, section-2, section-3) as well as the topic name.


Welcome Alex!

Logout

Choose one section section-3

Section: section-3; Topic Name: JQuery

Section: section-3; Topic Name: Vue

Section: section-3; Topic Name: React

You can click the drop-down list to show a list of options (sections) besides "all". By clicking one of the options, the topic list will be re-rendered to contain topics belonging to the chosen section only. For example, when we select the section option "section-3", the topics list changes accordingly.

Lab Exercise

Part 1. Prepare the React App

Step 1. Create a new React App using "create-react-app" command

Launch a terminal. Go to your "lab7" directory and create a React app named "myreactapp" using the following commands:

```
cd YourPath/lab7
npx create-react-app myreactapp
```

Go inside the "myreactapp" directory just created:

```
cd myreactapp
```

Then launch the React App as follows:

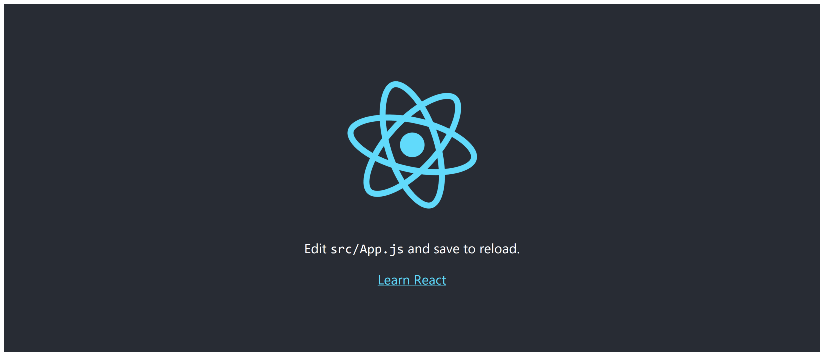
```
npm start
```

After successfully launching the app, you should see prompts like the following in your terminal:

```
Compiled successfully!

You can now view myreactapp in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://xxx.xxx.xx.xx:3000

Note that the development build is not optimized.
```

```
To create a production build, use npm run build.
…
```

And a web page should be loaded automatically in your browser, as follows:



Edit `src/App.js` and save to reload.

Step 2. Copy the provided files to myreactapp/src folder

Download and extract the files in “lab7_materials.zip”. You will see two JavaScript files “**App.js**” and “**index.js**”, and one CSS file “**index.css**”. Replace the original “App.js”, “index.css” and “index.js” files in **myreactapp/src** with the provided files.

Open “**index.js**” and compare it with the default one (page 10 of lecture notes 14_React_I_COMP3322A_f2022.pdf). The differences are:
Instead of

```
import App from './App';
```

we have

```
import Lab7App from './App';
```

And in the render() function, we replace “<App />” with “<Lab7App />”.

“Lab7App” is the React component which renders the main page, as implemented in “**App.js**”. Open “**App.js**” and check out the render() function of the class “Lab7App”. You will see it conditionally returns some components: If the state variable “**isLogin**” is false, which means that the user hasn’t logged in, the function returns a <div> containing 3 elements/compoment:

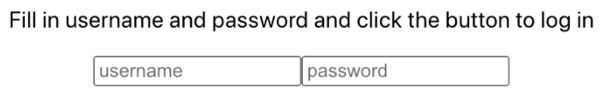
1. A <p> element displaying the message “Fill in username and password and click the button to log in”
2. Two <input> elements for username and password inputs.
3. A <LoginButton> component rendering the login button and achieving corresponding functionalities.

If the state variable “**isLogin**” is true, which means that the user has successfully logged in, the function returns a <div> containing 3 elements/component:

1. A <p> element displaying the message “Welcome xxx!”, where xxx is the name of the logged-in user.
2. A logout <button> element
3. A <Contents> component displaying the drop-down list and the list of topics.

You can refer to the screenshots at the beginning of this document to understand more of the components.

In this lab, we will only modify "**App.js**" to achieve the web page. Before you modify "**App.js**", the initial page is as follows when you launch the app using "npm start":


Fill in username and password and click the button to log in
username
password

Part 2. Implement the login and logout functionalities.

Step 3. Handle input changes in the two input boxes

In "App.js", class component Lab7App has a state of four key-value pairs: (1) *isLogin* is a boolean variable indicating whether the user has logged in; (2) *loginName* is a string variable containing the name of the logged-in user; (3) *inputUserName* is updated with the value of the username input box; (4) *inputUserPswd* is updated with the value of the password input box. By using **handleInputChange()** function as handler of the onChange event on the input boxes, values of the state variables *inputUserName* and *inputUserPswd* are always in sync with values of the input boxes. We have also provided the implementation of function **handleInputChange()**, where we use **this.setState{[name]: value}** to set value of a state variable according to the respective input value, where *name* refers to the value of the name attribute of the input element and *value* refers to the input value. Learn more about the usage of [name] at https://reactjs.org/docs/forms.html ("Handling Multiple Inputs" section).

Step 4. Implement the login and logout functionalities

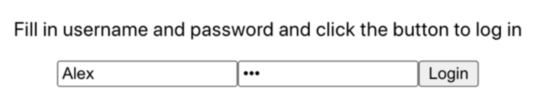
4.1. In the render() function of Lab7App, remove "{/* step 4.1 */}", and in its place, render the LoginButton component. The LoginButton component receives three props:

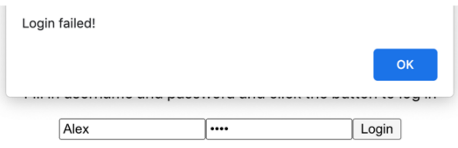
1. *username*: which equals the value of state variable *inputUserName*;
2. *password*: which equals the value of state variable *inputUserPswd*;
3. *handleStatusChange*: which equals *this.handleStatusChange*. We have provided the implementation of function **handleStatusChange()**, which receives the login status (boolean type) and the name of the logged-in user (string type) as arguments, and set the state variables *isLogin* and *loginName* to the argument values, respectively.

4.2. Go to class LoginButton. In its render() function, we are rendering a login button with the handler function *this.handleClick* bound to the onclick event. Implement the function **handleClick()**: If the input username and password are valid, use **this.props.handleStatusChange(xx, yy)** received from class Lab7App to update the state variables of Lab7App, where xx equals "true" and yy equals *this.props.username* received from class Lab7App; otherwise, alert the message "Login failed!" **Hint**: you can use the provided function **checkValid()** to check if the input username and password are valid, which returns true if the inputs are included in the predefined array *userDB*, and returns false, otherwise. Read more about the array find() function at https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find.

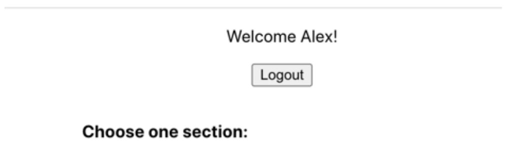
4.3. Go to class Lab7App. In the render() function, remove "{/* step 4.3*/}" and add an onClick event handler to the logout button, which is an arrow function *()=>this.handleStatusChange()* where *handleStatusChange()* receives a “false” value and an empty string as arguments.

After this step, when you click the login with incorrect passwords, you will see pages as follows:


Fill in username and password and click the button to log in

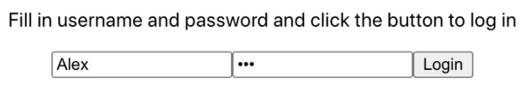

Login failed!

when you successfully login:


Welcome Alex!

Choose one section:

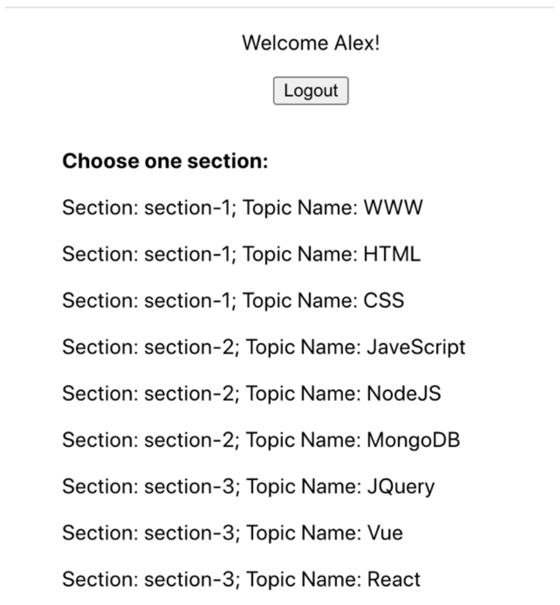
then clicking logout will lead you to the initial page:


Fill in username and password and click the button to log in

Step 5. Render topics list in class component Contents

Go to class Contents, which has a state of one key-value pair: *topics* stores the list of topics to be shown on the page, initialized using the given array *topicsDB*. Complete the render() function of class Contents: within **{ }**, apply **map()** function on *topics* to create a number of <p> elements, each presenting information of one topic in the *topics* array as <p>Section: {topic.section}; Topic Name: {topic.topic}</p>. The handler function **handleTopicsChange()** is to change the state variable *topics* upon selecting a different option in the drop-down list (to be implemented later).

After this step, you will see the topics list shown up on the page:



Welcome Alex!

Logout

Choose one section:

Section: section-1; Topic Name: WWW

Section: section-1; Topic Name: HTML

Section: section-1; Topic Name: CSS

Section: section-2; Topic Name: JaveScript

Section: section-2; Topic Name: NodeJS

Section: section-2; Topic Name: MongoDB

Section: section-3; Topic Name: JQuery

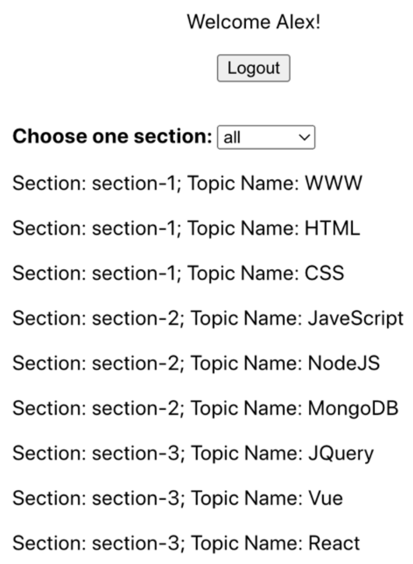
Section: section-3; Topic Name: Vue

Section: section-3; Topic Name: React

Step 6. Render a drop-down list in function component DropDownList

Now go to function DropDownList, which receives the handler function **handleTopicsChange** of class Contents in its props. Complete its render() function as follows: remove "{/* step 6*/}", and in its place, create a drop-down list using the <select> element with four options (with values “all”, “section-1”, “section-2” and “section-3”, respectively). The option “all” should be associated with the attribute “selected” as the default option. Add an event handler function to onChange event on the <select> element (to track the change of selected option and update topics list accordingly), which should be *e => props.handleTopicsChange(e.target.value)*.

After this step, you will see the drop-down list above the topics list, but the topics list will not change according to different options selected yet.


Welcome Alex!

Logout

Choose one section: all

Section: section-1; Topic Name: WWW

Section: section-1; Topic Name: HTML

Section: section-1; Topic Name: CSS

Section: section-2; Topic Name: JaveScript

Section: section-2; Topic Name: NodeJS

Section: section-2; Topic Name: MongoDB

Section: section-3; Topic Name: JQuery

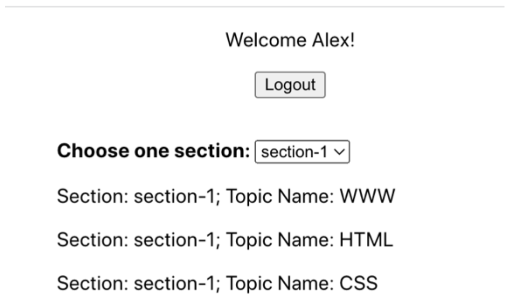
Section: section-3; Topic Name: Vue

Section: section-3; Topic Name: React

Step 7. Implement filtering of topics list

Now implement the function **handleTopicsChange()** in class Contents as follows: if value of the argument *option* is not "all" (i.e., it is one of the three sections instead), apply **filter()** function on *topicsDB* to obtain a filtered topics array where each topic's section is *option* (learn more about JavaScript Array filter method at https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter), and set the state variable *Topics* to be the filtered array; otherwise, set the state variable *Topics* to be *topicsDB*. Note that you need to use **this.setState()** to change a state variable's value.

After this step, when you select one section from the drop-down list, the topics list will change accordingly:


Welcome Alex!

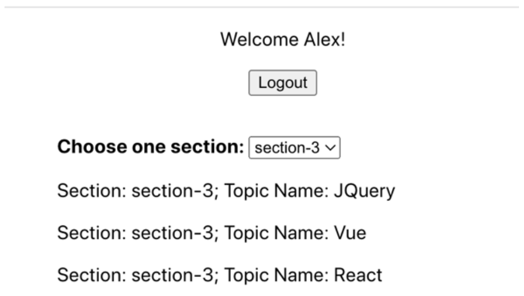
Logout

Choose one section: section-1

Section: section-1; Topic Name: WWW

Section: section-1; Topic Name: HTML

Section: section-1; Topic Name: CSS


Welcome Alex!

Logout

Choose one section: section-3

Section: section-3; Topic Name: JQuery

Section: section-3; Topic Name: Vue

Section: section-3; Topic Name: React

Congratulations! Now you have finished Lab 7. You should test the page and the final results should look similar to the screenshots at the beginning of this document.

Submission:

Please finish this lab exercise before 23:59 Sunday November 27, 2022. You should submit the **App.js** file only.

(1) Login Moodle.
(2) Find “Labs” section and click “Lab 7”.
(3) Click “Add submission”, browse your file and save it. Done.
(4) You will receive an automatic confirmation email, if the submission was successful.
(5) You can “Edit submission” to your already submitted file, but ONLY before the deadline.