# CMPS 356 Enterprise Application Development - Spring 2019

# Lab 11 – React JS Advance

## Objective

The objective of this lab is to practice how to secure web applications. You will practice:

- The React Context API's
  - React.createContext
  - Context.Provider
  - Context.Consumer
- React Routing
  - URL Prameters
  - 404 Pages
  - Nested Routes
  - Programmatically Navigating Users
  - How to protect our routes by creating Login/Logout Scenario
  - How to use Link and NavLink
- How to use Local Storage: client-side data storage to save frequently used data into the client machine such as user login tokens
- Optimize code with the user of
  - useReduce
  - Custom Hooks

## Overview

This lab has two parts:

• **Part A:** The Banking Application (1.5h).

• **Part B:** The BookStore Application (1.5h).

## PART A –Banking App

- Open the Banking app inside Lab 11 and run npm install to get all the dependencies.
- Install the **nodemon** package globally if you do not have it already.

  *npm install --save -g nodemon*

- To Run your application type in your terminal **nodemon** instead of "node app.js"

  ***Nodemon***

  Once you run your application using **nodemon**, the **Nodemon** will watch the files in the directory in which **nodemon** was started, and if any files change, nodemon will automatically restart your node application

1. Open the client directory inside your project
2. Create the **AddTransaction** component inside the **src/components** directory
3. Add the following Routes to the App Component

| Routes | Component to Load |
|---|---|
| / | Redirect to the **/accts/list Route** |
| **/accts/:action** | Accounts Component |
| **/accts /addTrans** | AddTrans Componenet |

4. Add the following routes to the Accounts Component

| Routes | Component to Load |
|---|---|
| **/accts/add** | AccountForm for adding account |
| **/accts/list** | AccountTable component |
| **/accts/:id** | AccountForm for [Editing an Account] |

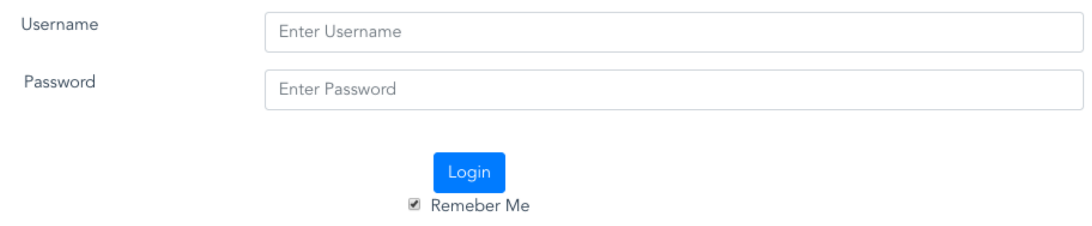
5. Modify the **AccountForm** component to accommodate the editing of an account. The **AccountForm** should receive an account as a prop.
6. Create the following **NavBar** and connect them to their corresponding components
   - **Accounts Link**=> should go to the **/accts/list** route
   - **AddTransaction** Link=> should go to **/accts/ addTrans**

7. Style the **active navigation link** to be green
8. Create the following links inside the Accounts Table
   - **Edit** => [Should be a link next to the delete button]
9. Restructure your **accountsTable** and move the row part to its own component named **Account**. You should pass all the necessary props to that component.
10. Inject the Account component back to your **AccountsTable to** render the accounts
11. Create the following component Login. To save time , you can get the login HTML code from the old repos
12. Create a registration component that allows users to register

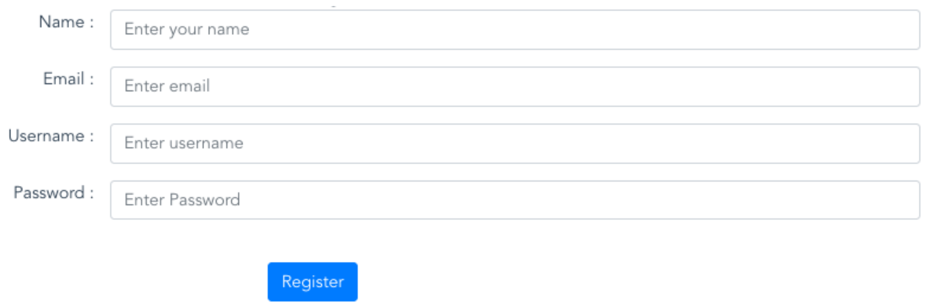
    **Note**: In the next lab we will implement this functionality on the server side however, for the demo of this lab, we will create our users locally and save them inside the local storage
13. Now add the following two routes to the App component and also add them to the **NavBar** component as **NavLinks**

| Routes | Component to Load |
|---|---|
| **/login** | Login Component |
| **/register** | Register component<br>➔Save the user data inside the local storage |

## Login Component


Username
Enter Username
Password
Enter Password
Login
Remeber Me

## Registration Component



14. When the application starts Initialize a state called **user,** by reading the local storage user account
15. Pass the **user object** to all the children components using the **Context API**

**16.** Protect all the routes that start with **/accts/: action**

- The first thing you need to do is check if the user is at the login or at the registration. If they happen to be in one of this routes then NO need to check if the user is logged in or not.

- Otherwise, you should check if **user** object is available. If it is then navigate the user to that component. **redirect** the user to the login screen.

17. Add **LogOut** button that clears the local storage

**PART B –BookStore APP**

Using Similar techniques as PART A, protect the BookStore app routes. Allow only logged in users to be able to add/remove/edit /borrow books

You can make use of the HTML, JavaScript and CSS provided in Lab 10 model solution.

You need to test your implementation as you progress and document your testing. After you complete the lab, fill in the ***Lab11-TestingDoc-Grading-Sheet.docx*** and save it in **Lab11-React.js-Advance** folder. Sync your repository to push your work to Github.