CS 3744 Spring 2020 - Assignment 5 - REACT (200 points)

Due : 5.7.2020 (Thursday) 11:59PM

Description

In this assignment, you will be given a REACT project file and a node.js server file. **This assignment is very similar to assignment 4.** Your task is to write a javascript file (App.js) that interacts with the Node.js server and add functionality to the buttons and the ToDo List UI. This will require you to understand essential components in React: how to structure functional components, how to use props and states, how to handle events, and how to render each component depending on the states and properties.

The project needs an 'App.js' With that file correctly in place, the ToDo List should perform and have the functionality as shown in the video attached/provided in this document.

Note that the index.html includes two external libraries (jQuery and Bootstrap). Given this is an external URL, it will function correctly only if you have an online connection. If you would like to have a local copy, feel free to download each file and revise HTML. However, you will be asked to only submit App.js and we will test your javascript file with the same environment.

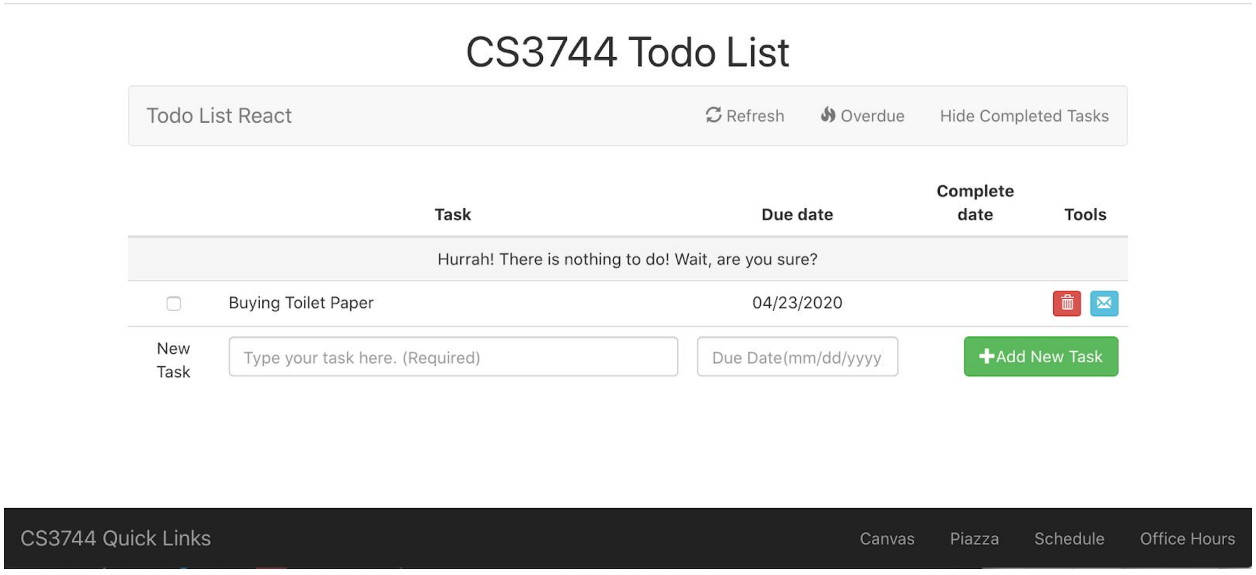
Even though you don't have to write a single line in Node.js part. You would have to READ server.js file thoroughly to understand how it works and what it asks for and how it returns. It is going to take a while to install the React app and set up the server. **START EARLY!**

Prerequisite

- Many of the functionality specifications will come from the specification of Assignment 4 (link). There are a few new features (a new task input form, a default message when there is no task in the list) in this assignment and there are a few features that are not included (Delete Completed Task button, Updating the task, no note). Therefore, the current specification will have lots of overlap with assignment 4 but neither a subset nor a superset of the assignment.
- The App.js, will be generated as part of the create-react-app node package (link). It is going to take some time to even set up the project.
- (Demo video for Assignment 5) - use VT account
    - https://drive.google.com/open?id=1lM2q_-R8O-J4SBl6jnsSjKKcrA96ixRR
- App.js file will provide a structure of components. YOU MUST follow the given structure. (WE WILL CHECK THE CODE.)
- I highly recommend you to check the key of Assignment 4.

Creating a REACT Boilerplate project.

- This assignment assumes that you have installed node.js and mongo database from assignment 4.
- Follow the tutorial here to install a default create-react-app project. (link)
- Install following node module packages
  - npm install mongodb --save
  - npm install express -- save
  - npm install jquery -- save
- Add the following line in the package.json (This is to run the server.)
  - "proxy": "http://localhost:8080",
  - Your package.json should look like this by now. (The version may vary.)
- Download server.js (--> this file is different from Assignment 4), index.html, App.js, and App.css from Canvas (Canvas > Files > Assignments > Assignment 5) (link). Place the server.js file in the project folder and replace the existing App.js, App.css, and index.html in the src and public folder with the ones downloaded from canvas. Keep the rest of the files in src and public untouched.
- You will need up to 4 command line terminals (minimum 3) to run the following programs in the project folder.
- Create a folder for your data for mongod --dbpath ./[foldername] (e.g. mongod --dbpath ./data if the created folder is "data"). This is same as the assignment 4.
- You will need to run the four following server programs.
  - **mongod --dbpath ./data** (maybe this is running already)
  - **mongo** ⇒ mongo console (optional) This link will be useful.
  - **npm start** ⇒ localhost:3000 (React program)
  - **node server.js** ⇒ localhost:8080 (This is for us to run the node.js server. For more detail see this link).
- Open your web browser and visit http://localhost:3000/#. You should see the following page.



CS3744 Todo List

	Task	Due date	Complete date	Tools
	Hurrah! There is nothing to do! Wait, are you sure?			
☐	Buying Toilet Paper	04/23/2020		
New Task	Type your task here. (Required)	Due Date(mm/dd/yyyy		Add New Task

Specifications

1. **Compliance** (50pts)
    a. For this assignment, we will check App.js to make sure that the students are using exactly the provided functional components. (Do not merge them. Do not create any new ones.)
    b. jQuery is provided only for AJAX calls. You. For those who want to explore further, one can use fetch API (a javascript native alternative to jquery AJAX call) instead of jQuery AJAX call.
    c. The web page should never hang (or freeze) in any case, you will lose points (up to 50) for a web page that hangs due to an infinite loop in react.
    d. You are **NOT** allowed to have any global variables or functions other than functional components. The only exception to this is two helper functions given (getFormattedDate, reviveDate). Each case of global variables will be penalized. Note that typically incorporating every functional component in App.js is a bad practice. We are doing this just for the sake of convenience of submission and grading. In practice, you should have a separate file (or folder) for each component, in which case you cannot really have global variables.
    e. By the way, if you submit the downloaded App.js as is, you will get 50pts from this section, and of course, you will get 0 from the rest of the specification.
    f. (Hint) You will have to modify existing JSX code e.g. adding props, adding event handlers, etc.
    g. (Hint) Monitor server.js terminal frequently to check messages. Putting the fetchdata in the wrong place will end up creating an infinite loop. (link)
2. Create New Task (20pts)
    a. Using the input form on the bottom of the page, one should be able to create a new task.
    b. You should do the sanity check in the exact same way you did in Assignment 3.
    c. (Hint) Use react states to link input forms with variables (useState). Once you get an Okay sign from newtask endpoint request, the program should fetch data to reflect the change.
    d. Before you implement fetching data you should be able to check if new data is added from the mongo console. Check this website(link) to find out how to use Mongo Shell commands and the Piazza post (link).
3. Rendering Tasks from the server. (30 pts)
    a. As it is getting data from the server, it means that you will get the same set of task items even after you refresh the page and even after you restart the server.
    b. Note that _id field is automatically generated by the Node.js app. Use it to update tasks later.

   c. (Hint) TodoList is the best place to fetch the data as one can generate an array of TodoItem components based on the array you get from the server, using javascript array map function (link) - of course, you can do in your own way (e.g. for-loop). And many other components (e.g. NavBar, NewTask, TodoItem) should be able to tell the TodoList component to fetch data again when something is updated (e.g. refresh button is clicked, a new task is added, a task is completed, when a task is completed, etc.). You can create a function inside TodoList (e.g. fetchData) and give the function handle to its children so that they can request. Fetching data should only happen 1) in the beginning, when the component is rendered, 2) when necessary. useEffect hook should be helpful for this (link).
   d. The server will respond with task data as an array in the order of created date (from old to new).
   e. You do not have to truncate the title this time (because we don't have Note any more).
   f. Check out the assignment 4 specifications for more details. (overdue task should be in red.)
4. Rendering when no task (10pts)
   a. When there is no task to complete, it should display a message (an example is given in the starter code).
   b. Conditional rendering should be useful.
5. Refresh button (30 pts)
   a. Refresh button will fetch data from the server and re-render everything.
   b. For example, if I open the website in a new tab and add a task (or update a task) and the refresh button on the other page should get the most up-to-date data from the server.
6. Completing a task (15 pts)
   a. You do have to update the model on the server and fetch new data.
   b. Check out the assignment 4 specifications for more details. (green color, crossing out the title).
7. Deleting a task/Emailing a task (15 pts)
   a. You do have to update the model on the server and fetch new data.
   b. For email icons, you do NOT have to worry about the note. Make sure that you encode your text with the encodeURI function.
   c. Other than that, Check out the assignment 4 specifications for more details.
8. Overdue (15 pts)
   a. Pressing the overdue button shall make only tasks that are overdue present in the table.
   b. (Hint) Your Navbar has to talk to its sibling (TodoList). The only way to communicate with the sibling is to talk to its parent (App) - that is lifting up states - because global variables are NOT allowed. This is same for Hide Completed Tasks button (9)
   c. (Hint) filter function of javascript array will be useful (link) but there can be other ways to do it (e.g. for-loop).

    d. You do not have to update the model on the server. Just change the view locally.
    e. Users should be able to “toggle” the button (active when it’s on)
    f. Other than that, Check out the assignment4 specifications for more details.
9. Hide Completed Tasks (15 pts)
    a. You do not have to update the model on the server. Just change the view locally. Using Hide Completed Task button, a user can filter out completed tasks
    b. Users should be able to “toggle” the button (active when it’s on)
    c. Check out the assignment 4 specifications for more details.
10. Extra Points (30pts)
    a. Implement an update button for each item using inline editing (no modal window). Make this part of App.js and indicate that you implemented extra parts in Canvas. (5pts)
    b. Add sorting function; by clicking the header of each column, one should be able to sort the existing table in the ascending/descending order of due date, task title, complete date. You don’t have to synchronize the sorting state with another computer and it is okay to lose the state when you refresh the page. Make this part of App.js and indicate that you implemented extra parts in Canvas. (5pts)
    c. Make each to-do list item drag-and-droppable (example) so that one can reorder them in any way that they want. Make this as part of App.js and indicate that you implemented extra parts. You don’t have to synchronize the reordered state with another computer and it is okay to lose the state when you refresh the page. Make this as part of App.js and indicate that you implemented extra parts in Canvas. (10pts)
    d. Running this on a web server that we can access via URL. (5pts) ⇒ Do not share this with anybody in the class (now and in the future). This needs to be taken down at the end of the semester (Give us the URL.)
11. Honor Code Violation
    a. If you share any outcome of Assignment 1, 2, 3, 4, 5 with anybody in any form, it will be a violation of the honor code. I will periodically crawl the Internet, GitHub repositories, and personal websites.
    b. I have reported a number of students for honor code violations in the past, for your information. Feel free to use the code to demonstrate your capability but do it at your own risk (do it locally, temporarily, or as a video).

Notes

- You are going to submit ONLY one JS file (App.js) so do NOT revise the HTML file and do NOT revise server.js. We are going to grade with the HTML file and the server.js of ours (the one that is available in the Canvas) with the JS file that you submit.
- We are going to grade in the most recent Chrome.
- For me, it took 297 lines to complete the work.
- Note that the video is also part of the specification. (like Assignment 3 and 4).
- This is the first time that we teach REACT in this class. This means that GTAs/UTAs did not learn React from the previous courses they took. Please be patient even in case they

cannot immediately answer your question. Please use me (Piazza) and I will do my best to answer promptly. Also, ask me for a separate appointment if you need additional support. Otherwise, I will run office hours more often than usual. (Look out for announcements in Piazza).

- NO LATE WORK will be ACCEPTED (except the case of using late days if you have any left). Do NOT email me.