CS 3744 Spring 2020 - Assignment 4 - Server Interaction (150 points)

Due : 4.14.2020 (Thursday) 11:59PM

Description

In this assignment, you will be given an HTML file and a node.js server file with a pre-styled ToDo list. This assignment is almost exactly the same as the assignment 3. Your task is to write a javascript file that interacts with the Node.js server and add functionality to the buttons and the ToDo List UI. This will require you to understand what elements/containers are on the page, how to make AJAX requests, and how the server takes requests and sends responses in order to make the program correctly function.

The HTML File needs an 'index.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 index.js and we will test your javascript file with the given index.html.

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 node.js and run mongodb. START EARLY!

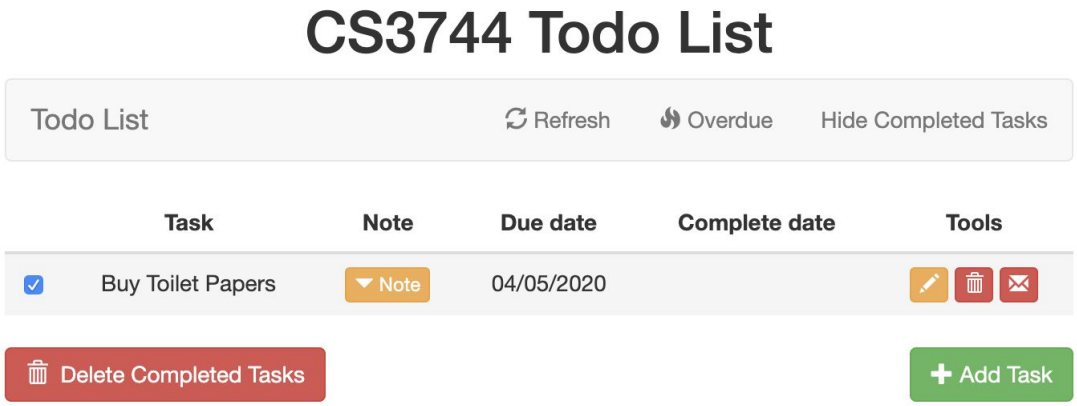
Prerequisite

- Many of the functionality specifications will come from the specification of Assignment 3 (link). There are a few new features in this assignment (Refresh, updating the task). Therefore, the current specification will be a super set of the specification of Assignment 3.
- The index.js, index.html has to be put in a specific folder (public) and you would have to learn how to run node.js file. I recommend you to watch the lecture recording of 4/1 before starting the assignment.
- (Demo video from Assignment 4) - use VT account
    - https://drive.google.com/open?id=1cW2xt6sW4KYo24NaNG_SmpYceQshfSRU
- (Demo Video from Assignment 3) - use VT account
    - https://drive.google.com/file/d/1aHQTlNvxzR-YuHq6pa26U87zfRds38aq/view?usp=sharing
    - https://drive.google.com/open?id=1aXOiOtRUeyrk3XyZdM4cyN8RwmeaKJul

- Given index.js file will provide some structure. I recommend you to follow the given structure although you do not have to.
- I highly recommend you to check the key of Assignment 3.

Install Node.js and mongodb:

- Follow the tutorial here to install node.js in your computer. https://nodejs.org/en/download/
- Download Node.js installable setup from its official site and install with default options (LTS). After installation, you will be able to run node and npm on the command line.
- Install mongo database https://docs.mongodb.com/manual/installation/
- Create a folder for assignment4. Navigate to this folder and in a command prompt (Terminal, Command, PowerShell). If you do not know how to navigate in a command prompt program, please see the following pages
    - (Linux/Mac) https://www.macworld.com/article/2042378/master-the-command-line-navigating-files-and-folders.html
    - (Window) https://docs.microsoft.com/en-us/windows/nodejs/beginners, https://www.digitalcitizen.life/command-prompt-how-use-basic-commands or https://programminghistorian.org/en/lessons/intro-to-powershell#moving-between-directories-with-set-location-sl-cd
- Create a node app following the tutorial inside the folder that you created (link). As part of this tutorial, you will be asked to install the express npm package and create a “public” folder.
- Install mongodb package inside the folder by running npm i mongodb --save
- Replace the server.js with the one available on Canvas (Canvas > Files > Assignments > Assignment 4) (link)
- Download index.html and index.js from the canvas folder to the public folder.
- Create a folder for your data, open another Terminal window and run mongo database using mongod --dbpath ./[foldername] (e.g. mongod --dbpath ./data if the created folder is “data”)
- Run node server.js on another terminal window.
- You can open another window and run mongo to monitor your database and collections.
- Open your web browser and visit http://localhost:8081# and see if you can get the index.html. You should see the following page.



Functionality Expected:

1. Creating a new task (25pts)
    a. Note that I got rid of form tags from the html pages so you don't have to worry about default behavior.
    b. Before you implement fetching data you should be able to check if new data is added from the mongo console. Check this website to find out how to use Mongo Shell commands (link) and check out the demonstration video to see how I access the database.
2. 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 task later (4, 6, 7, 11). Note that the fetched data does not have id (index number that starts from 0) anymore. However, you can still use index to access the tasks array.
    c. The server will respond with tasks data as an array in the order of created date (from old to new).
    d. Check out the assignment 3 specification for more details
3. Refresh button - New! (10 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.
4. Updating a task (20 pts)
    a. The yellow pencil button right next to the delete task button will open another modal window.
    b. One should be able to update any content (title, note, due date) of a task using the button.

    c. There is a hidden input element that you can use for storing _id of the task that is loaded and sending _id for the server. (line 143, index.html file).
5. Expandable/collapsable note. (5 pts)
    a. Now you have a due date as part of the content. (new)
    b. Check out the assignment 3 specification for more details .
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 3 specification for more details.
7. Deleting a task (15 pts)
    a. You do have to update the model on the server and fetch new data.
    b. Check out the assignment 3 specification for more details.
8. Emailing a task (5 pts)
    a. Now, you DO have to worry about line breaks. Make sure that you encode your text with the encodeURI function.
    b. Other than that, Check out the assignment 3 specification for more details.
9. Overdue (5 pts)
    a. You do not have to update the model on the server. Just change the view locally.
    b. Other than that, Check out the assignment 3 specification for more details.
10. Hide Completed Tasks (5 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. When the hide completed tasks button is clicked, the delete completed task button should be disabled. (NEW)
    c. Other than that, Check out the assignment 3 specification for more details.
11. Delete Completed Tasks (15 pts)
    a. You do have to update the model on the server and fetch new data.
    b. Check out the assignment 3 specification for more details.
12. Extra Points (20pts)
    a. Note that the current example given has no pagination and it will be too much to fetch all data if there are too many items (e.g. more than 20)
    b. Fetch items no more than you need. (e.g. fetching 100 items and only show 20 items is not allowed)
    c. Use bootstrap pagination to separate pages.
    d. For this part, you have to revise the HTML file and the server as well. Please submit two additional files **SEPARATELY** in addition to index.js which will cover 1~11.
    e. Name files as follows: index-page.html, index-page.js, server-page.js. Pagination function in index.js will not be graded.

Notes

- You are going to submit ONLY one JS file (index.js) so do NOT revise the HTML file and do NOT revise server.js. We are going to grade with the HTML file and ther server.js of ours with the JS file that you submit.
- Make sure that you do not add any extra style (inline style) beyond using bootstrap.

- We are going to grade in the most recent Chrome.
- For me, it took approximately 300 lines to complete the work.
- Note that the video is also part of the specification. (like Assignment 3).
- Using jQuery is highly recommended but not required. You can use pure javascript.