Overview

For this assignment you need to make a database backed website that features Ajax interaction. You have almost two weeks to work on this assignment. Please start as early as possible. For the grading purpose, this assignment **doesn't allow late submission**.

At the end of this description you will find code which will create a handler you can visit to set up your database table. It contains all the fields needed to make a simple workout tracker.

- name - the name of the exercise
- reps - the number of times the exercise was performed
- weight - the weight of the weights used
- date - the date the exercise was performed (in the format of Month-Day-Year, e.g., 05-25-2018)
- unit - indicating if the measurement is in lbs or kg

Requirements

You need to create a **single page application** with the following functionality:

Visiting the page will show a table displaying all completed exercises. The header should list all the columns (id should not be displayed in the header or in the table itself. Use hidden inputs to keep track of the id).

At the top of the page there should be a form that let you enter in all the data needed to make a new entry in the table with a button to submit it. Hitting that button should add the row to the table if it was successfully added to the database. If it was not successfully added (probably because name was left blank and it is required) it should not add it to the table.

Each row should have two buttons. **One to delete the row and one to edit the row.** Hitting the delete button should immediately remove the row from the table and from the database.

Hitting the edit button should make it possible to edit the data. For this function it is OK to go to another page which will allow you to edit that specific exercise, save it and then take you back to the main page. The form to edit the exercise should be pre-populated with the existing data from that row (in other words if I go to the edit page, and then hit save, nothing should change because all the old values were in the edit form).

All interactions, other than updating an exercise, **should happen via Ajax**. This means that at no time should the page refresh. Instead **Ajax calls should be used to GET or POST to the server and it should use the data the server provides to update the**

page. (Think back to the AJAX interactions assignment, where you updated your page's DOM to show the response you received from OpenWeatherMap, for example)

(Links to an external site.)Links to an external site.

Here is an example of what I mean:

http://jsfiddle.net/GRgMb/ (Links to an external site.)Links to an external site.

If you hit delete on one of those rows it gets deleted without the page refreshing. Your page should do this and it should update the database at the same time to reflect the deleted data. It should essentially happen in reverse when you add a row. You hit add and the table is populated with a new row.

For the "single page", it means you should use one URL for both types of requests. You will have only one handlebars file which is home.handlebars. But in your server side code, you will write two methods:
1) app.get('/',.....)
2) app.post('/',.....)

These methods will make sure to call the get method when the client sends a GET request and call the post method when the client sends a POST request.

Helpful Suggestions

In this assignment, you need to implement both frontend and backend. If you finished all previous assignments, you should already have a solid foundation for frontend technologies and the frontend would not be a problem for you. The backend is more difficult than frontend and you need to implement your own API to manipulate mySQL. For example, in "Using SQL with Node", there are several examples of APIs to insert / select / update / delete data in mySQL.

Generally, you can think the html, css and javascript for html are frontend and they run in the browser and provide a UI for the webpage. Node.js and mysql are backend. Because Ajax allows the web page to be updated asynchronously, it is a frontend technology.

The data flow should look like the following.

Request: Ajax -> Node Server -> MySQL

Respond: mySQL -> Node Server -> Ajax

Returning Data From The Database

Because the interactions should be handled via Ajax, you often only want the database to send back an updated version of the table, not a whole new page. If you go back to

the very first example with Express.js in week 7, you will see an example where we return plain text rather than HTML in a fancy Handlebars template. You can use this same technique to return a simple JSON string (which conveniently is what is shown being displayed in the MySQL demos). Just send that back to the browser in response to an Ajax request and build a table using it rather than generating the HTML on the server.

You could even do this when you make the page initially. Just have JavaScript make an Ajax request when the page is loaded to get the JSON representing the table. You never even need to build a table on the server that way.

Handling the Update and Delete

Hidden fields will be key to deleting and updating data. Every row should have a form (or two froms) holding the update and delete buttons. That form should also have an input of type="hidden" which holds the id of the row so you can easily pass that information to the server to delete or update the row.

Including JavaScript

Include static JavaScript files just like statics CSS files. See the content about organization in week 8. It discusses how to link a static CSS file. You can do the same thing to link static client side JS files.

Grading

Not implementing Ajax interactions will limit your maximum grade to an 80%. They add a fair bit of complexity so if you meet all the requirements but nothing works via Ajax then you can still get 80% credit for the assignment.

Properly implementing the **viewing, editing, adding and deleting** of data will be worth 25% each.

Deliverables

Add an **URL to your site in the text box** during submission. As part of the assignment submission, we want you to include a link to where you have setup the script to run. So it will be the address in which we can access the site. For example, you setup and run on flip1 port: 3000, the address will look like **http://flip1.engr.oregonstate.edu:3000** The port will be the one you specified in the javascript, and the flip server is based upon which one you started the script on. It is your responsibility to keep it up and running until you receive a grade. Submit **a zip file containing all of your source code**. Please include your source files and your package.json file (no need to include the node_modules folder and files)

```
app.get('/reset-table',function(req,res,next){

  var context = {};

  [your connection pool].query("DROP TABLE IF EXISTS workouts",
function(err){ //replace your connection pool with the your variable containing the
connection pool

    var createString = "CREATE TABLE workouts("+
    "id INT PRIMARY KEY AUTO_INCREMENT,"+
    "name VARCHAR(255) NOT NULL,"+
    "reps INT,"+
    "weight INT,"+
    "date DATE,"+
    "lbs BOOLEAN)";

    [your connection pool].query(createString, function(err){

      context.results = "Table reset";

      res.render('home',context);

    })

  });

});
```

PreviousNext