ASSIGNMENT 3: USING DATABASE VIA NODE.JS

RELEASED: **4 NOV 2018**
DUE: **18 NOV 2018 23:59**

SYNOPSIS

You are going to store and manipulate some data in a database. Build a RESTful API to serve HTTP responses for creating, retrieving, and deleting data.

BASIC SETUP

During *lab 5*, you have learnt to set up a basic web server using ***Node.js*** and ***Express***. Using either local Node.js installation or online Node.js playgrounds like Runkit.com, set up a web server which would listen on port 3000.

In *lab 6*, you have further set up a ***MongoDB*** database, and access it using ***Mongoose*** in Node.js. It is all right to use local MongoDB installation or the MongoDB Atlas Cloud.

DATABASE SCHEMA (20%)

Schemas and models in Mongoose help to confine the types of data to be stored. These two schemas are needed in this assignment:

Event

- **`eventId`**: number, required, unique
- **`name`**: string, required
- **`loc`**: *ObjectId referring to Location documents**
- **`quota`**: number

Location

- **`locId`**: number, required, unique
- **`name`**: string, required
- **`quota`**: number

* The Event schema has been defined in lab 6, but some amendment is needed to accommodate the reference to documents in the ***Locations*** collection.

GET AND POST (30%)

Your Express server ought to respond to these requests:

- **`GET http://`***server address***`/event/`***event ID*

  When the server receives this request, it should look up the event with the given event ID from the database. Then simply print the event name, location ID, location name, event quota on separate lines. The location quota is *not* necessary. If the given event ID is not found, output an understandable message. All the response should be sent as HTTP responses, to be shown in the web browser. *(Done in lab 6 but improvement needed)*

- **`POST http://`***server address***`/event`**

  When the server receives this request, it should use the parameters submitted in the HTTP request body to create a new event in the database. A simple HTML form like this can be used: *http://www.cse.cuhk.edu.hk/~chuckjee/2720lab6/form.html*

However, instead of letting the user decide the event ID, your code should look into the database to find the maximum current event ID, e.g. ***x***. Assign ***x+1*** as the new event ID. The form should ask the user for the location ID only but not the location name. A lookup should be done to check if the location quota is larger than or equal to the new event quota. If not, the event should not be created and an error message should be responded to the user.

If the event is created successfully, respond to the user with an appropriate message (*e.g., "Event created!"*), together with the event details as in the **`GET`** response. The HTTP status code should be 201 here.

DELETE (20%)

To support the RESTful API, we would allow users to delete an event using this request:

- **`DELETE http://`***server address***`/event/`***event ID*

  If the event ID is found, the event should be removed from the database. An appropriate message is shown to the user, with the event details as in the **`GET`** response. If the event ID is not found, show an understandable error message to the user.

You may utilize the Chrome Application ***Postman*** for making **`DELETE`** requests to your Express server during your development. Of course, you may also build user pages to send it using JavaScript. This JS is not part of the assignment submission.

SOME MORE QUERIES (30%)

The following requests should also be entertained:

- **`GET http://`***server address***`/event`**

  List all the events available, with details in separate lines.

- **`GET http://`***server address***`/loc`**

  List all the locations available, with details in separate lines.

- **`GET http://`***server address***`/loc?quota=`***number*

  List all the locations with quota of at least this number. Show an appropriate message if there is none.

- **`GET http://`***server address***`/loc/`***location ID*

  Show the details for this location ID. Show an error if the location ID is not found.

- **`GET http://`***server address***`/event/`***event ID***`/loc/`***location ID*

  List all the events which has this event ID, or which are held at this location ID.

- **`POST http://`***server address***`/loc`**

  Create a new location with the input from a user form. Like events, decide the location ID based on the current maximum location ID.

BONUS FEATURE

The **`PUT`** request for a complete RESTful API is still missing. Implement a convenient user interface for ***updating*** any event and location information. The current data from the database should be shown to the user, so that they can decide what to edit. Then update the database with the user data, using these requests:

- **`PUT http://`***server address***`/event/`***event ID*
- **`PUT http://`***server address***`/loc/`***location ID*

Show a message when done. You may submit an extra HTML file for this task for the form.

This task is NOT a requirement of the assignment. But if you successfully do it, it contributes to a few bonus points for your course grade:

- *Complete fulfilment = **0.5 points***
- *Partial fulfilment = **0.25 points***

FEATURES AND MODULES

There is no cosmetic requirement for this assignment. You are welcome to implement extra styles and features at your own ability. Besides ***Express*** and ***Mongoose***, you can use any Node.js module at your discretion, but ***please only use one JS file.***

SUBMISSION

Before submitting, you may feel free to try with your own data in your database. We will test your work using our database server, so you can safely clear the database server URL in the JS file for submission.

We will only visit your web page submission using ***Google Chrome*** (almost-latest versions) and the Chrome Application ***Postman***. Please utilize the ***Developer Tools*** for debugging of your code.

Please read this article carefully: *http://www.cuhk.edu.hk/policy/academichonesty*

Include your full name and student ID in ***all code files*** using comments. Zip all your files and submit it on the course site on Blackboard. ***Only these files should be included:***

- 1 `.js` file
- A number of HTML files, named appropriately to show their use
- `package.json`/`package-lock.json` from Node.js

You do not need to submit the* `node_modules` *folder.