CSCI 2720 – Building Web Applications

Assignment Three: Database

Released: 23rd November 2023

Due: 23:59, 14th December 2023

Synopsis:

You are going to store and manipulate some data in a MongoDB database. The data are related to events and locations, similar to the suggested dataset in our project.

Environment setup:

Please follow ***lab07*** to set up a basic web server using Node.js and Express. Also, please follow ***lab08*** to set up a MongoDB database and access it using Mongoose in Node.js.

Problem 1: Database Schema (20%)

Schema and Model in Mongoose help to confine the types of data to be stored. Your task is to set up two Schemas:

1. **Event:**
    a. ***eventId*****:** *number, required, unique*
    b. ***name*****:** *string, required*
    c. ***loc*****:** *ObjectID -> location documents**
    d. **quota:** *number*
2. **Location:**
    a. ***locId:*** *nmuber, required, unique*
    b. ***name:*** *string, required*
    c. ***quota:*** *number*

*Remarks: Try to accommodate the reference to documents in the **location** collection. Check Lecture 17 MongoDB from the Blackboard.

Problem 2: GET (15%)

Your Express server should respond to this request:

GET http://server-address/ev/eventID

When the server receives this request, look up the event with the given event ID from the database. Then print the event *name, location ID, location name, event quota* on separate lines, as shown in the illustration below. The location quota is not necessary. All field names and string values should be quoted with double quotes. Extra spacing is allowed, and brackets/commas do not need to be in separate lines.

If the given event ID is not found, output an understandable message in the response body with **status code 404**. All the response should be sent as HTTP responses with content type **`text/plain`**.

```
{
"eventId": 123,
"name": "CSCI2720 lab",
"loc":
{
"locId": 1,
"name": "SHB924"
},
"quota": 2
}
```

Problem 3: POST (15%)

Your Express server should respond to this request:

POST http://server-address/ev

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 ***lab7_task2.html*** in Blackboard can be used.

However, instead of letting the user decide the event ID, your code should assign a new maximum event ID by looking into the database. For example, if 123 is the current maximum event ID, the new event ID would be 124.

Modify the form so that it asks the user for the location ID. 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 with **status code 406**. (Note: There is no need to check other event at the same location.)

If the event is created successfully, respond to the user with the URL of the created event. Use HTTP **status code 201**.

Problem 4: DELETE (10%)

We allow users to delete an event using this request:

DELETE http://server-address/ev/eventID

If the event ID is found, the event should be removed from the database. Send a response with **status code 204**, and nothing in the body. If not found, show a simple error message in the body with **status code 404**.

Problem 5: Some more queries (20%)

The following four requests need to be handled:

Q1: GET http://server-address/ev

List all the events available, with details formatted similar to the illustration below.


```
[
{
"eventId": 123,
"name": "CSCI2720 lab",
"loc":
{
"locId": 1,
"name": "SHB924"
},
"quota": 2
}
,
{
"eventId": 124,
"name": "CSCI2720 lab 2",
"loc":
{
"locId": 1,
"name": "SHB924"
},
"quota": 2
}
]
```


Q2: GET http://server-address/lo/locatoinID

Show the details for this location ID, with details formatted similar to the illustration below. Respond with an error message if the location ID is not found with the **status code 404**.


```
{
"locId": 1,
"name": "SHB924",
"quota": 100
}
```

Q3: GET http://server-address/lo

List all location available. Show details similarly to the illustration in Q1.

Q4: GET http://server-address/ev?q=number

List all the events with quota, with a similar format in Q1. Respond with an empty array if there is none.

Problem 6: Updating with PUT (20%)

Implement a simple HTML user interface for updating event information. Assume that the user always types in an existing event ID to load the event information. The current data from the database should be shown to the user, so that they can decide what to edit, in an HTML form. Then, update the database with the user data on submission, using this request:

PUT http://server-address/ev/eventID

The event should always be updated successfully. Respond to the user with the updated event details as in Problem 1. You may combine this HTML form with the POST form in Problem 3 or create a new HTML file. Checking of event and location quota is not necessary here.

Formats and assumptions:

Please pay attention for the followings:

1. In this assignment, since the response body do not always confirm to standard JSON formats, e.g., for error messages, please always use **`text/plain`** as the **`Content-Type`**.
2. The JSON-like objects and arrays should be readable by standard JavaScript **JSON.parse()** if all newline characters are removed.
3. Use the CORS middleware for express to enable submission from local HTML forms (more details in ***lab07***).
4. Unless otherwise specified, the HTTP status code should be **200 OK** by default. **201 Created** is used for responding to successful POST requests, and **204 No Content** is used for responding to successful DELETE requests.
5. For development, you may feel free to populate your database with testing data (i.e., create your own fake data for testing). Please create a folder named ***database_setup***. You can create database whose name is ***assignment3*** and create two collections which are locations and events. In the locations collection, please import from JSON file ***database_setup/assignment3.locations.json***. In the events collection, please import from JSON ***database_setup/assignment3.events.json***.
6. For grading, graders will use another set of data in their database. Hence, you code should be adaptive to different data. Assume that there is always at least one location and one event in the database when grading (i.e., it will not be an empty database). All locations linked from events are valid in the database.
7. Assume that the input data type is always correct, i.e., only numbers will be input for number fields. All numbers are positive numbers. When grading, no fields are blank.
8. Assume that user always enters a valid and existing location ID when creating or updating events.
9. Unless otherwise specified, events or locations with missing quota will not be tested.
10. The Node server should start at port 3000. Assume the access URL starts with http://localhost:3000
11. We will not test situations not mentioned in this document.
12. There are no cosmetic requirements for this assignment. You are welcome to implement extra styles and features at your own ability.
13. Your submission should contain only one JavaScript file.

Submission:

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

Plagiarism is heavily penalized. Never share your code. Please read this article carefully: https://www.cuhk.edu.hk/policy/academichonesty/

Include your full name and student ID ***on top of all code files using comments***. You should also include the required ***code header for honesty declaration***. Check that the file names and formats are correct. Zip all your files and submit it on the course site on BlackBoard. You can have unlimited submission. We will only grade you latest zip file. Name your zip file as:

`(SID)_asg3.zip`

Your zip file should only include:

1. One `.js` file
2. A number of HTML files, named appropriately to show their use (e.g., `problem_3.html`, `all_in_one.html`, etc).
3. `package.json` and `package-lock.json`
4. Your ***database_setup*** folder containing your own testing data (i.e., `assignment3.locations.json` and `assignment3.events.json`).

Important: Remove the *node_modules* folder.

If you need help, please feel free to contact the main TA for Assignment Two:

Mr. Liu Qixuan (qxliu@cse.cuhk.edu.hk or 1155203213@link.cuhk.edu.hk), AB1-403, Saturday from 8 a.m. to 5 p.m.

Or the course instructor:

Dr Colin Tsang (colintsang@cuhk.edu.hk), SHB130, by appointment.

Remark: We welcome any questions, but we will not do debugging for you.

Late submission: A 30%-mark deduction will be applied for late submissions within three days. Late submissions more than three days will not be graded.

Code header for honesty declaration:

```
<!-- I declare that the lab work here submitted is original
except for source material explicitly acknowledged,
and that the same or closely related material has not been
previously submitted for another course.
I also acknowledge that I am aware of University policy and
regulations on honesty in academic work, and of the disciplinary
guidelines and procedures applicable to breaches of such
policy and regulations, as contained in the website.

University Guideline on Academic Honesty:
https://www.cuhk.edu.hk/policy/academichonesty/

Student Name : <your name>
Student ID : <your student ID>
Class/Section : <your class/section>
Date : <date> -->
```