SFWRTECH 4WP3 – Advanced Web Programming - Assignment #6

Weight

4% of total course grade

Due date

Monday March 14th at 11:59pm

Primary learning objectives

- Create a RESTful API with Node.js
- Use SQLite to manage a database table

Requirements

Create a RESTful API using Node.js and SQLite.

Your RESTful API should manage a collection of properties data. Use the initdb.js file in the assignment starter code zip file to create an SQLite database and table of properties data. All of the columns of data in this database should be made accessible for the below RESTful API operations. The primary key (rowid) in your SQLite table will be the same id used by the RESTful API.

The RESTful API should be available at **localhost:3000/api**. i.e. it should be running on the localhost, on port 3000. And the api should be accessible at **/api**. You can assume that the initdb.js scrip has been run to setup the table of properties before your API server code is executed.

Your RESTful API must allow for the following operations:

- A GET request to **/api** should return the entire collection of data as a JSON array of JSON objects in the body of the response.
- A GET request to **/api/id** should return the item with the supplied id as a JSON object in the body of the response.
- A PUT request to **/api** should replace the entire collection with the collection provided as a JSON array in the body of the request. The API should return the JSON response **{response: "COLLECTION UPDATED"}**.

- A PUT request to **/api/id** should replace the item in the collection with the supplied id with the item provided as JSON data in the body of the request. The API should return the JSON response **{response: "ITEM UPDATED"}**.
- A DELETE request to **/api** should delete the entire collection. The API should return the JSON response **{response: "COLLECTION DELETED"}**.
- A DELETE request to **/api/id** should delete the item in the collection with the supplied id. The API should return the response **{response: "ITEM DELETED"**}.
- A POST request to **/api** should insert into the collection the item provided as JSON data in the body of the request. The API should return the response **{response: "ITEM INSERTED"}**.

Create a test script to test your API. At a minimum, the test script should attempt to do the following in this order:

1. Insert multiple items into the collection (either with a couple POST operations and/or the PUT collection update operation).
2. Delete at least one item from the collection.
3. Get the collection that results from these operations. Use console.log to output the resulting collection to the console so that you can check to see the data is as expected.

As part of the above sequence of requests, the test script should make a minimum of 6 requests to the API to ensure the API is functioning correctly (e.g. insert 4 items, delete 1, and a final get request to check the results…. or insert 3 items, delete 2 items, and a final get request to check the results, etc.).

You will need to make HTTP requests to the API (which you can assume is running), to execute these tests. Use axios and async/await to make these requests complete in the order above (i.e the GET request should not occur until the Delete request has responded).

If you want to make a more sophisticated test script, this is welcome. You could for example test each operation, and ensure that the body of each response returns what is expected (either something like "ITEM INSERTED" or in the case of GET operations, check the collection/item is as expected). However, this is optional, a simpler test as above is sufficient.

Your RESTful api file should be called **server.js** and your test script should be called **test.js**.

Submission

Zip your solution as assignment6.zip and upload it to the dropbox on Avenue.

Marking rubric

Component	Description	Marks
RESTful API	Required 7 operations	/50
Test code	Test code with 6 requests	/30
axios	axios used to implement test code correctly	/20
	Total:	/100