CMPS 350 – Web Applications Design and Development

Assignment 04 – Web API

Instructions

1. The deadline for completing your work is **Sunday, April 16, 2023 @ 12:00 PM**.
2. No late submission will be accepted.
3. Push your code to your private repository under `assignments/assignment04`.
4. Commit often and use meaningful message summaries and descriptions.
5. Any form of plagiarism will be treated seriously and reported.

1. Next.js Web API [50 points]

Create an `assignment04/words-app` directory then create a Next.js 13.3 application inside `words-app` that implements the following API to manage multiple collections of words.

1. Data is provided to you in `words.json`. The file contains an array of almost every word in the (American) English language. Copy that file to `words-app/data/words.json`.
2. Add an endpoint to search for a certain number of words using a query text. The number of matching words is selected **at random** and returned. Return 10 words when the number of words is not provided. You should return at most 100 words per request.
3. Add an endpoint to read all collections. This endpoint is also used to create collections. Store your collections and their corresponding words under `words-app/data/collections.json`. A collection is a set of **unique** words:

```
[
   { name: collection_0, words: [word_0_0, word_0_1, …, word_0_m] },
   { name: collection_1, words: [word_1_0, word_1_1, …, word_1_n] },
   …
]
```

4. Add an endpoint to read all words in a collection, add a word to a collection, and delete a word from a collection.
5. You should validate all parameters provided by the user to the API and respond with an error (`4xx`) for invalid requests, such as an invalid parameter value, attempting to delete a non-empty collection, or attempting to delete a word from a collection that does not contain it.
6. You should handle all server errors and respond with an error status and message (`500`).
7. You should respond with correct status codes for successful requests (`2xx`).

Method	URL	Description
`GET`	`/api/words/:query&count=:count`	Returns an array of words, with size count, all containing the string passed in the query URL parameter.
`GET`	`/api/collections`	Returns an array of all collections.

`POST`	`/api/collections`	Creates a new collection. The name of the collection is part of the request body.
`GET`	`/api/collections/:name`	Returns an array of all words in the collection with the provided `name`.
`DELETE`	`/api/collections/:name`	Deletes an **empty** collection with the provided `name`.
`POST`	`/api/collections/:name/:word`	Adds the provided `word` to the collection with the provided `name`.
`DELETE`	`/api/collections/:name/:word`	Deletes the provided `word` from the collection with the provided `name`.

2. Testing using Postman [10 points]

Test every route and handler in the API using Postman then export the Postman collection of tests as `postman.json` and commit it along with your solution.

3. Client-side Application [40 points]

Create a client-side application that provides a user interface to manage multiple collections of words using the API developed earlier.

1. Create an HTML file under `words-app/public/index.html`. You can access and test this page using `http://localhost:3000/index.html`.
2. Allow the user to list existing collections, create a collection, and delete an **empty** collection.
3. Allow the user to select a collection and list all its words. The user should be able to add a word to the selected collection based on the words resulting from a search query.
4. Allow the user to delete a word or move it from one collection to another.