Name: Shrey Shah (N01549811)

Raghav Bedi

Project

Step 1: Loading the "Sample Restaurant Data" in MongoDB Atlas ‘

A screenshot of a computer

Description automatically generated

Step 2: Building a Web API

• You need to install express, cors, mongoose + set up Git repository

A screenshot of a computer program

Description automatically generated

• Add a module to interact with Restaurant MongoDB (Similar to assignment 3)

o "Initializing" the Module before the server starts ▪ To ensure that we can indeed connect to the MongoDB Atlas cluster with our new connection string, we must invoke the db.initialize(“connection string…”) method and only start the server once it has succeeded, otherwise we should show the error message in the console

A black screen with many small text

Description automatically generated with medium confidence

o This module will provide the 6 (promise-based) functions required by our Web API for

this particular dataset

o db.initialize("Your MongoDB Connection String Goes Here"): Establish a connection with

the MongoDB server and initialize the "Restaurant" model with the "restaurant"

collection (used above)

A screen shot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

o db.addNewRestaurant(data): Create a new restaurant in the collection using the object

passed in the "data" parameter

o db.getAllRestaurants(page, perPage, borough): Return an array of all restaurants for a

specific page (sorted by restaurant\_id), given the number of items per page. For

example, if page is 2 and perPage is 5, then this function would return a sorted list of

restaurants (by restaurant\_id), containing items 6 – 10. This will help us to deal with the

large amount of data in this dataset and make paging easier to implement in the UI

later. Additionally, there is an optional parameter "borough" that can be used to filter

results by a specific "borough" value

o db.getRestaurantById(Id): Return a single restaurant object whose "\_id" value matches

the "Id" parameter

o updateRestaurantById(data,Id): Overwrite an existing restaurant whose "\_id" value

matches the "Id" parameter, using the object passed in the "data" parameter.

o deleteRestaurantById(Id): Delete an existing restaurant whose "\_id" value matches the

"Id" parameter

A screenshot of a computer program

Description automatically generated

Add the routes : The next piece that needs to be completed before we have a functioning Web

API is to actually define the routes (listed Below). Note: Do not forget to return an error

message if there was a problem and make use of the status codes 201, 204 and 500 where

applicable.

• POST /api/restaurants

o This route uses the body of the request to add a new "Restaurant" document to the

collection and return the created object / fail message to the client.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

• GET /api/restaurants

o This route must accept the numeric query parameters "page" and "perPage" as well as

the string parameter "borough", ie:

/api/restaurants?page=1&perPage=5&borough=Bronx. It will use these values to return

all "Restaurant" objects for a specific "page" to the client as well as optionally filtering

by "borough", if provided.

o EXTRA CHALLENGE (bonus): add query param validation to your route in order to make

sure that the params you expect are present, and of the type you expect. You can do

this using packages like https://www.npmjs.com/package/celebrate or https://expressvalidator.github.io/docs/check-api.html . If the params are incorrect, your route should

return a 400 response (client error) vs. 500 (server error).

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

• GET /api/restaurants

o This route must accept a route parameter that represents the \_id of the desired

restaurant object, ie: /api/restaurants/ 5eb3d668b31de5d588f4292e. It will use this

parameter to return a specific "Restaurant" object to the client.

A computer screen shot of text

Description automatically generated

A screenshot of a computer

Description automatically generated

• PUT /api/restaurants

o This route must accept a route parameter that represents the \_id of the desired

restaurant object, ie: /api/restaurants/5eb3d668b31de5d588f4292e as well as read the

contents of the request body. It will use these values to update a specific "Restaurant"

document in the collection and return a success / fail message to the client.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of text

Description automatically generated

• DELETE /api/restaurants

o This route must accept a route parameter that represents the \_id of the desired

restaurant object, ie: /api/restaurants/5eb3d668b31de5d588f4292e. It will use this

value to delete a specific "Restaurant" document from the collection and return a

success / fail message to the client.

A computer screen with text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated