**ITE5315 Project**

**Project Team:** Team of two

**Assessment Weight:** 20% of your final course Grade

Help full link : [Get Started with Atlas — MongoDB Atlas](https://www.mongodb.com/docs/atlas/getting-started/)

Sample restaurants :L [Sample Restaurants Dataset — MongoDB Atlas](https://www.mongodb.com/docs/atlas/sample-data/sample-restaurants/#std-label-restaurants-restaurants)

Helpful videos:

<https://www.youtube.com/watch?v=RcxdF3Lzoac>

<https://www.youtube.com/watch?v=sx3Lf2EaEEQ>

<https://www.youtube.com/watch?v=EOq6KU2BOYY>

<https://www.youtube.com/watch?v=ChiC7zhhQzQ>

<https://www.youtube.com/watch?v=9YxgKmO-Rlw>

https://www.youtube.com/watch?v=4qTAPKhTItU

**Assessment includes:**

 Final project submission includes all project file + project document

**Description:** Using the concepts that we learned in the course and, we are going to develop a secure db-driven Node/Express app.

**Project Specification:**

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

**Step 2:** Building a Web API

 You need to install express, cors, mongoose

 Add a module to interact with Restaurant MongoDB

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

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)

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

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.

 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.

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.

 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.

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.

**Step 3:** Pushing to Heroku Once you are satisfied with your application, deploy it to Heroku

**Step 4 :** Use Environment Variable for your Connection String Your solution currently has your database connection string (with username and password!) hard coded into your source code. This is a potential security risk. If this code is shared between users on a team, or if it is pushed to a public repo on GitHub, your password is now public too.

ProjectSubmission:

 Add the following declaration at the top of .js files /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* ITE5315 – Project \* I declare that this assignment is my own work in accordance with Humber Academic Policy. \* No part of this assignment has been copied manually or electronically from any other source \* (including web sites) or distributed to other students. \* \* Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student ID:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \*

**LATE SUBMISSIONS for assignments**. There is a deduction of 10% for Late assignment submissions, and after three days it will grade of zero (0).