ITC5315: Project Status Report WK 1

Created by: <Bahare Ghasemi>

Student ID: <N01538197>

Date: <2024-04-11>

# Table of contents

[Table of contents 2](#_Toc736272803)

[Step 0 3](#_Toc2003563819)

[Step 1 6](#_Toc920770620)

[Step 2 8](#_Toc411941608)

[Summary 25](#_Toc433382319)

## Step 0

Describe why you selected to use a git repository or not:

I prefer to use git because I need to have an experience on Git and additionally it provides below advantages:

* I have a full history of my code and changes.
* I can work in a team that can be so big.
* Switching to any previous state of project would be easy.
* Branches provide a separate space to work on each different feature.
* A full copy of all the history of the project will be available on every machine. Consequently, no code will be lost.
  1. Create new repository in Git and then create git init:

A screenshot of a computer program

Description automatically generated

* 1. Add repository as remote:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. Create a readme file, commit it to repository, push it to github:

A screenshot of a computer

Description automatically generated

* 1. Observe readme file in the github:

A screenshot of a computer

Description automatically generated

## Step 1

Show a screenshot Atlas/Compass once the data has been loaded. Use the Mongosh terminal to run the command: db.<collection>.countDocuments() on the “restaurants“ collection in the “5315-project” database. Show a screenshot of the result. (see <https://www.mongodb.com/docs/manual/reference/method/db.collection.countDocuments/>)  
**Atlas:**

A screenshot of a computer

Description automatically generated

**Compass** **with document command**:

A screenshot of a computer

Description automatically generated

## Step 2

Use “npm run start” to start your server. Show a screenshot of it being started using this command.

Explain any challenges or new learnings that you encountered when implementing the routes. Use Postman to test all the routes. Make sure to include tests for errors / invalid data. List the response codes that you used and why you used them. Show screenshots of all your tests.

Challenges:

* Initializing Module before start server: This part had challenges to understand the question. I learnt how to start my connection before running app.js. Other parts were like previous assignments, and I reviewed them to do some parts such as input validations.
* Below part was the most difficult part for me. Validation took more time than I expected. Program stopped, and did not work properly, so I made it functional to make sure program will not stop in any situation.

A screen shot of a computer program

Description automatically generated

**2.1. Add start to script:**

A screenshot of a computer program

Description automatically generated

**2.2. Restaurant Module:**

A screenshot of a computer program

Description automatically generated

**2.3. Initializing Module with 6 async functions in database.js:**

**2.3.1. Connecting to database, add new restaurant item to database:**

**A screen shot of a computer program

Description automatically generated**

**2.3.2. Retrieve all restaurants items, retrieve one restaurant item by Id, Update one restaurant item by Id:**

**A screen shot of a computer program

Description automatically generated**

**2.3.3. Delete one restaurant item by Id, exports all functions:**

**By exporting functions from the module, these are accessible to app.js.**

**A screenshot of a computer program

Description automatically generated**

**2.5. Adding Routes and npm run start:**

**2.5.1 Firstly, I installed express-validator:**

**A screenshot of a computer program

Description automatically generated**

**2.5.2 npm run start: Initialize connection and then run.**

1. **POST /api/restaurants:**

**A screenshot of a computer program

Description automatically generated**

1. **GET /api/restaurants**:

**A screenshot of a computer program

Description automatically generated**

1. **GET /api/restaurants:**

**A screen shot of a computer program

Description automatically generated**

1. **DELETE /api/restaurants:**

**A screen shot of a computer program

Description automatically generated**

1. **PUT /api/restaurants:**

**A screenshot of a computer

Description automatically generated**

# **Tests**:

**If I changed .env file, app.js does not run and response error in console:**

**A screenshot of a computer

Description automatically generated**

1. **Post new item to restaurant:**

**It did not add restaurant\_is because I typed the name of column wrong.**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Then I updated by put:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. **Getting all restaurants by accepting inputs:**

**If I don’t pass any params:**

**A screenshot of a computer

Description automatically generated**

**If I pass input by invalid type:**

**A screenshot of a computer

Description automatically generated**

**If I don’t pass borough:**

**A screenshot of a computer

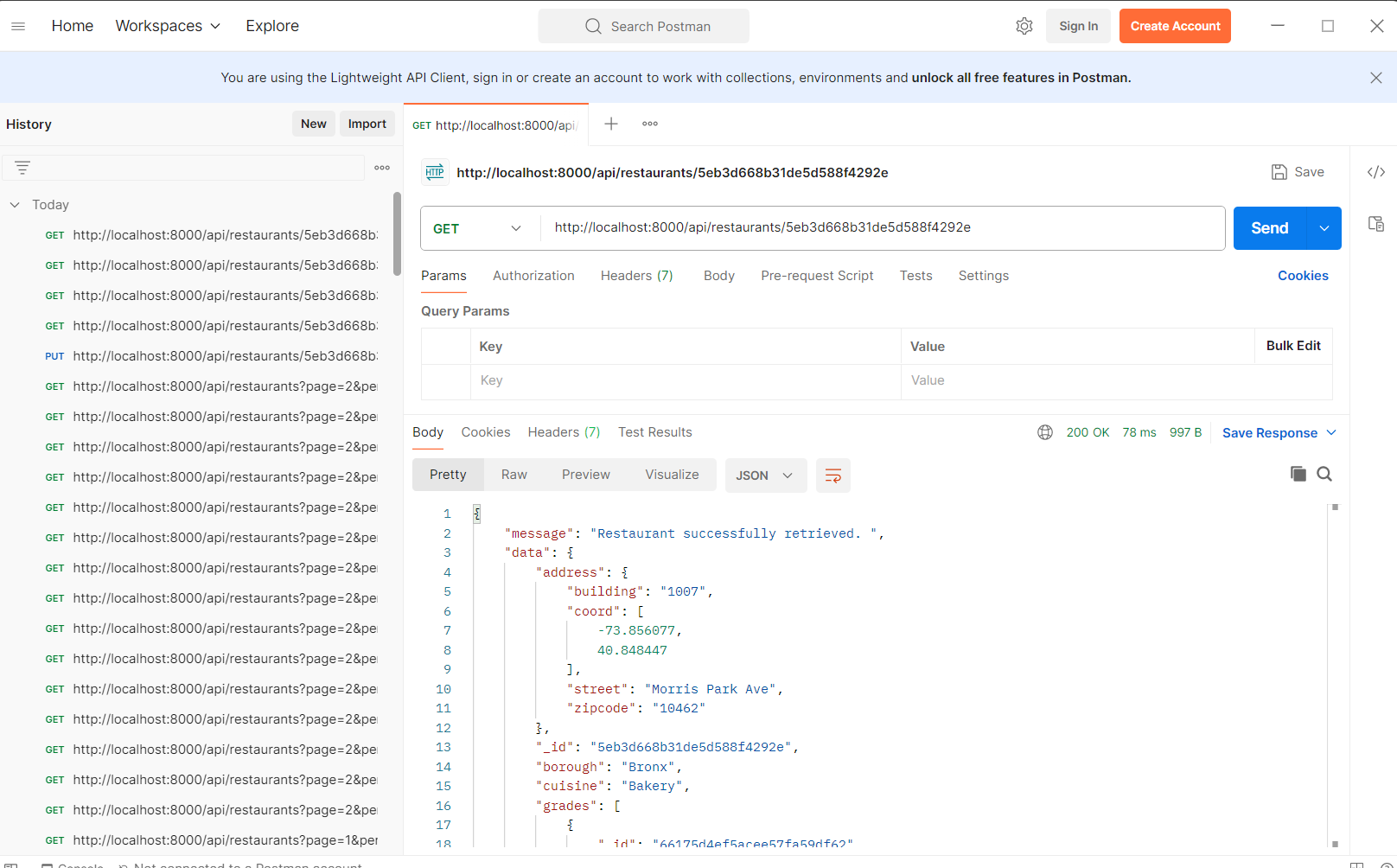
Description automatically generated**

**By passing borough:**

**A screenshot of a computer

Description automatically generated**

1. **Get restaurant by Id of an object:**

****

**When id is not valid:**

**A screenshot of a computer

Description automatically generated**

1. **Update an item by Id:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**When id is not valid:**

**A screenshot of a computer

Description automatically generated**

1. **Delete an item in restaurant:**

**A screenshot of a computer

Description automatically generated**

**When id is not valid:**

**A screenshot of a computer

Description automatically generated**

Step 3

Describe the steps you used to create the UI. Describe any challenges or new learnings that you encountered while developing the UI. Include screenshots of the UI and screenshots of Postman being used to test the new route.

Steps:

1. Setup Handlebars as the Template Engine: I used Handlebars as the template engine using app.engine('.hbs', engine({extname: '.hbs'})); and app.set('view engine', '.hbs');.
2. Create Search route: I created a route (/api/restaurants/search) to render the search form (restaurantForm.hbs) by get.
3. Design a form: I designed restaurantForm.hbs with input fields for page number, items per page, and borough. I used HTML input elements such as <input type="number"> and <input type="text">.
4. Handle form submission: I created a post route (/api/restaurantForm) to handle the form submission. In this route, I used express-validator to validate the form inputs. If there were validation errors, I rendered the error.hbs template with error messages. I retrieved restaurant data by input and rendered the restaurantDisplay.hbs template with data.

In this part, I had some challenges with the handlebar’s routes and setting, but I checked previous assignments.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Added parts to app.js:

A computer screen shot of a program

Description automatically generated

## Summary

Summarize your experience while working on this project for the first week. Do you have questions about the project for the next class? Questions you want to ask your peers or professor? Would you have done anything differently? Did anything take shorter/longer than you expected?

During the first week of working on this project, I faced several challenges and had a valuable learning experience:

1. MongoDB Interaction: Writing database functions to do create, get, update, and delete. Learning how to connect to the database and run server initially and execute queries.
2. Review express: Since this project involves building an API using express, I had to review my assignments and get more familiar with this technology. Understanding how to structure routes, handle requests, interact with the database, and implement error handling was crucial.
3. Middleware and Validation: Learning how important middleware functions are in express for input validation using the express-validator library was challenging. I had to understand how to define validation rules for different situations and handle validation errors gracefully.
4. Error Handling: Error handling logic for different situations such as validation errors, database errors, and server errors and showing the correct http status codes and error messages to be meaningful to user.
5. Testing with postman: Testing both valid and invalid inputs for each route and make sure that the API responses as expected in various situations.

To summarize, I should mention I liked this assignment more than others and I reviewed all subjects which I learnt in this course. The defining routes and testing were the easiest part. On the other hand, the error handling and template engine were the challenging part which I spent most of the time on and it was more than I expected.