ITC5315: Project Status Report WK 1

Created by: Meenu Mathew

Student ID: N01582144

Date: 7-04-2024

# Table of contents

[Table of contents 1](#_Toc736272803)

[Step 0 2](#_Toc2003563819)

[Step 1 3](#_Toc920770620)

[Step 2 5](#_Toc411941608)

[Summary 6](#_Toc433382319)

## Step 0

Describe why you selected to use a git repository or not

I decided to use the Git for my project as it is a complete package including a safety place to store the codes, version control, collaboration between teams and backup and recovery.

I used git to mainly track my project progress because in each commit it gives insight into the changes I had made. Every commit is like a snapshot of the current project at that time. Hence it helps to ensure data integrity and minimize data loss.

Also, a version control helps to divide project between different phases like week1 and week2.

Main advantage of git is it helps to collaborate between team members if different members are working on the same project, though it is a single project, it helps to share code between members for debugging of errors.

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/>)

A screenshot of a computer

Description automatically generated

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.

A screenshot of a computer program

Description automatically generated

For the proper working of the application validation added to both POST and PUT method. The parameters ‘name’, ‘cuisine’ and ‘restaurant\_id’ are made as required.

For searching restaurants using page, perPage and borough parameters, validation was not properly working for POST route as I was using only body and validationResult.

const { body, validationResult } = require('express-validator');

Then I changed it to const { body, param, query, validationResult } = require('express-validator');, then validation worked for GET routes also.

Learned to add pagination in express.js code and sorting with different parameters.

return await Restaurant.find(query).skip(skip).limit(perPage).sort({ restaurant\_id: 1 });

**Response codes:**

**500: Internal Server error**

Server encountered a situation, where it cannot resolve.

**400: Bad request**

The server will not process the request due to client error.

**201: Created**

The request successfully executed, and a new resource was added.

**200: OK/Success**

Any successful request

**Get all Restaurants**

A screenshot of a computer

Description automatically generated

**Validation for query parameter exist and correct**

A screenshot of a computer

Description automatically generated

**Get Restaurant By ID**

A screenshot of a computer

Description automatically generated

**Test with non-existing restaurant id**

A screenshot of a computer

Description automatically generated

**Add new Restaurant**

A screenshot of a computer

Description automatically generated

**Validation check when adding new restaurant**

A screenshot of a computer

Description automatically generated

**Update a restaurant**

A screenshot of a computer

Description automatically generated

**Validation check when updating a restaurant**

A screenshot of a computer

Description automatically generated

**Delete a restaurant**

A screenshot of a computer

Description automatically generated

**Delete test with non-existing restaurant id**

A screenshot of a menu

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.

**Design steps:**

1. Installed Express-handlebars package
2. Public folder with images and stylesheets created
3. Views folder created and added view pages
4. Inside the route view page render option provided.

Handlebars views (search\_restaurants.hbs and index\_restaurants.hbs) are used to render the form and results respectively.

‘express-handlebars’ package is used to create the UI for the application. An image is added into public/images folder and a **css** file added to public/stylesheets folder. The view pages are added in **views** folder.

Learned to take values from array values to display. Here in the table the address of the restaurant is taken from an array.

Learned some styling techniques that can be used in html tables.

Handlebars was showing some error when I tried to access parameters from database. To solve that I used **runtimeOptions** in the app.engine settings.

runtimeOptions:{allowProtoPropertiesByDefault:true,

      allowedProtoMethodsByDefault:true},

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

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 project I concentrated on the basic structure of the project.

1. Initialized git repository to store all the codes.
2. Understood the requirements and divided them into different parts.
3. Created new Node.js project and installed necessary dependencies. Express.js for routing, Mongoose for mongodb database integration, express-validator for validation.
4. Implemented all the routes/api endpoints in app.js.
5. Created a restaurant model to connect api points with database, to add, update and fetch restaurant details.
6. An initialize function added to initiate server only when mongodb connected to the application.