ITC5315: Project Status Report WK 2

Created by: MALIKA GUPTA

Student ID: N01581424

Date: April 15th 2024

# Table of contents

[Table of contents 1](#_Toc1122846617)

[Step 3 2](#_Toc557882305)

[Step 4 3](#_Toc499316495)

[Step 5 4](#_Toc1509694456)

[Step 6 5](#_Toc1563392119)

[Summary 6](#_Toc1613282810)

## Step 3

Describe your process for building the UI and Form support using a template engine. Describe which fields you provided in the form. Describe the CSS/styling that you applied. Describe any challenges you faced and how you approached solving them. Show screenshots of your UI. Show screenshots of testing the new route using Postman

For UI and form, I used express-handlebars. The main.hbs file in views->layouts provides the basic template for all webpages of this app, and custom webpages for different routes such as homepage, login, sign up etc. have its own .hbs file which adds data to the {{body}} section of main.hbs!  
  
Forms for both, sign up and login use ‘username’ and ‘password’ fields. These are validated from frontend as well as backend to ensure protection from data injection.

A screen shot of a computer code

Description automatically generated

Form for page, perPage and borough information to display restaurants based on these filters use 3 fields, page, perPage and borough. Borough is optional. These form fields are validated from frontend for required/optional constraints and they are validated from backend for datatype as well as required/optional constrains.

A computer screen shot of a program code

Description automatically generated

CSS applied is quite standard and simple. The css file is linked in the main.hbs file from which it is adding styles to all webpages. Added color and different font colors to header, footer to separate them from the webpage’s body. Added flex display, padding, spacing, background color to the navigation bar so that it looks organized.

Homepage for a logged in user:

A screenshot of a computer

Description automatically generated

Form for restaurant details(GET /api/restoform):

A screenshot of a menu

Description automatically generated

(POST /api/restoform):

A screenshot of a computer

Description automatically generated

Login page:

A screenshot of a menu

Description automatically generated

Signup page:

A screenshot of a computer

Description automatically generated

Login process with non-existing username – handled with proper tailored error message:

A screenshot of a menu

Description automatically generated

A screenshot of a computer error

Description automatically generated

Login process with wrong password:

A screenshot of a computer error

Description automatically generated

Signup using an existing username:

A screenshot of a computer error

Description automatically generated

**Challenges faced with UI:**

Frontend CSS related challenges – wasn’t able to understand why my navigation bar at the top wasn’t displaying list options vertically. Resolved it by changing the nav display to flex and justify-content to space-between.

**Challenges faced with Backend:**

Took me a while to understand the best practices for error handling when the user makes a bad request but I figured it out and was able to give the right response codes as well as tailored error messages for all scenarios!

**POSTMAN for new routes:**

PUT: update a restaurant - only works when you have a cookie in the browser with correct value, which is the hashed \_id of a user from the mongoose users database.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Without cookie – redirected to the login page. This does not give us an error because we simply don’t want to give the PUT access to a non-user so we are directing them to login:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Errors when entered an invalid \_id:

A screenshot of a computer

Description automatically generated

Errors when entered a non existing \_id:

A screenshot of a computer

Description automatically generated

LOGIN(SUCCESS):

A screenshot of a computer

Description automatically generated

LOGIN (failure – no password):

A screenshot of a computer

Description automatically generated

LOGIN (failure – no username/invalid username):

A screenshot of a computer

Description automatically generated

LOGIN (failure – username password mismatch):

A screenshot of a computer

Description automatically generated

LOGOUT – the cookie disappears and we are redirected to the homepage:

A screenshot of a computer

Description automatically generated

LOGIN (Username does not exist in the database):

A screenshot of a computer

Description automatically generated

## Step 4

Describe your approach to providing security in your application – Did you make any changes to the database? Describe the changes you made to support authorization of users. Show screenshots of testing authorization using Postman or your Browser.

**Security was added to this application by several ways:**

1. Database collection to store username and password.

A screenshot of a computer

Description automatically generated

1. Storing encrypted password by using bcrypt library in the database. We create hashed passwords by using salt rounds. The value of salt rounds is stored in an .env file so that it is not accessible to everyone.

A screen shot of a computer program

Description automatically generated

1. Jwt and cookies – I used java web tokens to create cookies using the database \_id of user logged in to store their session information. The secret key is a crucial component here and we used it to sign the JWT so that it is not publicly revealed. Authentication and security of JWT is ensured as it can be verified that the token is generated by a trusted source and has not been tampered with. We use the Secret key to check the \_id in the cookie matches with our database user’s \_id and continue the session for a user. **Secret key is also stored in an .env file for proper security.**

Middleware function to check if there is a jwt cookie in the session and to further check if the token is valid

A screen shot of a computer program

Description automatically generated

1. Expiration of cookies and jwt – I ensured that a user login is protected by setting the cookie to expire after 1 day and the jwt token to also expire after 1 day. Also, the cookie is accessible for http requests only.

A computer screen shot of a code

Description automatically generated

A computer screen with many colorful text

Description automatically generated

## Step 5

Describe the new feature you added and why to choose that feature. Show screenshots of you demonstrating the new feature in your browser. Describe any challenges you faced with the new feature you added.

**New feature : Find your new favourite restaurants:** If you are a signed-in user of this app, you can filter restaurants by cuisine and borough to see restaurant name, address and grades! This way, you can find restaurants that serve the cuisines you like in a particular area.

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 menu

Description automatically generated

## Step 6

Describe any challenges you faced deploying to Vercel. Show screenshots of your application running on a Vercel domain. Provide the public URL of your application on Vercel (make sure it does not require authentication)

Tried a bunch – did not work for me! ☹

## Summary

Summarize your experience while working on this project for the second week. Would you have done anything differently? Did anything take shorter/longer than you expected?

This was a fantastic learning experience. Basically built the entire project using Assignment 4 as the foundation. YouTube was a great source to speed up the process and build on the concepts taught in class and assignments.

It took quite longer than expected because debugging was hard for me. Wish I had more time to make it look even prettier using CSS but I think my backend work was solid.