Student Name: Tay Kai Xiang Matriculation Number: A0200236Y

Link to GitHub repository: https://github.com/kaixiangtay/CS3219_OTOT_C

Pre-requisites (Software to be installed on local device):
Postman : https://www.postman.com/downloads/
NodeJS and npm: https://docs.npmjs.com/downloading-and-installing-node-js-and-npm
MongoDB: https://docs.mongodb.com/manual/installation/

Overview
For this task, I have used:

- bcryptjs to hash the user password during sign up and password equality check during login.
- jwt web token as a form of authentication while added an admin role to demonstrate the authorization for the FoodTracker backend Get Food api call where only admin with the correct json web token attached as authorization in the request header during API call will have access to all the food data in the application.

Step 1: Clone the project repository CS3219_OTOT_C into the desired directory of local device.

Note: In this repository, we have only 1 folder called backend which is build upon Task B and 1 postman payload file called FoodTracker_Postman.json with sample API calls testing for Task C.

Step 2: Open up the terminal /console window and go to the directory in Step 1. Then, inside the project folder of CS3219_OTOT_C execute following command to be inside the backend folder directory:

cd backend

Step 3: Install the node application dependencies as specify in the package.json file required by backend NodeJS application using the following command:

npm install

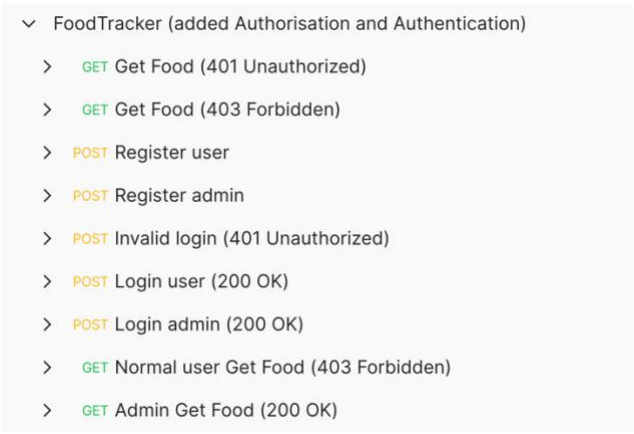
Step 4: Create the .env file in the root directory of backend folder and paste the following configurations:

JWT_ACCESS_TOKEN = 'hsdfhfyerfrfh125254345gfwdfe5d43243dghsdce7f4r839e3dyuedy7d7346etd8ucfhweuf734r4788sdiofuefn3948ejfhwerufhrf874537467344641fgsgdhsdchwdfg378es4834'
PORT = 8080
DB_URL = 'mongodb://localhost/FoodTracker'

Step 5: Next, start up the backend application using the following command:

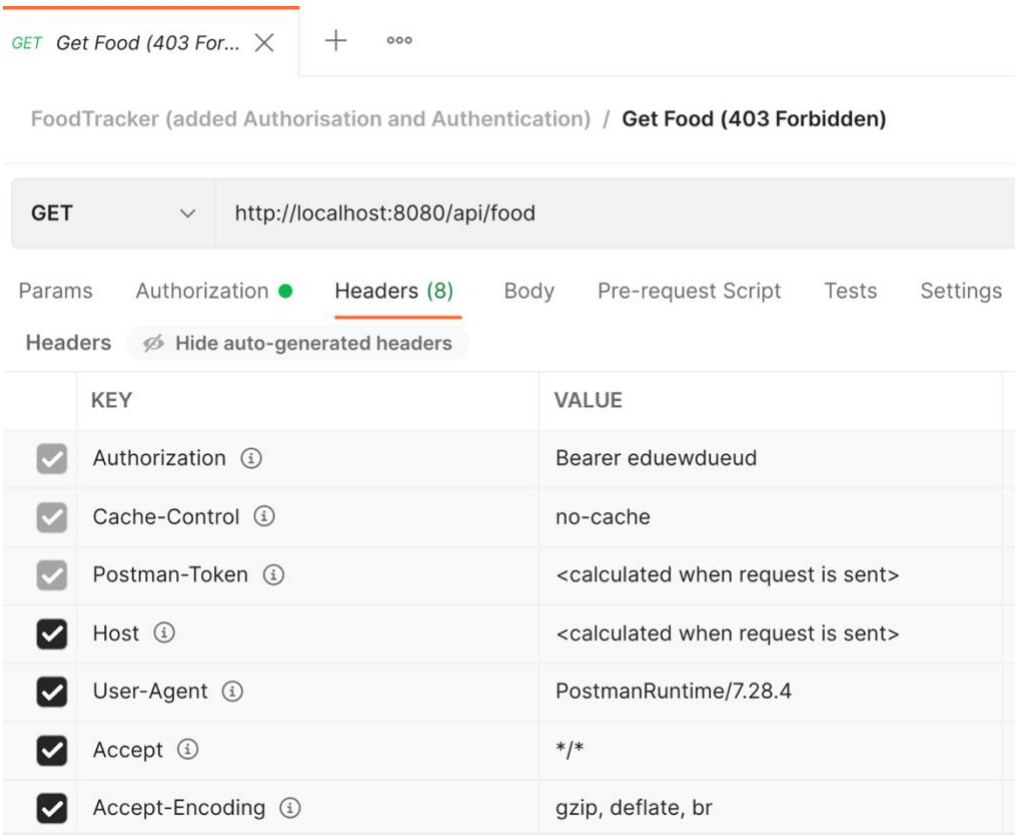
npm start

Step 6: In Postman software, import the api payload file, **FoodTracker_Postman.json.**



Step 7: We can go down the API list in order by running “Get Food (401 Unauthorized)” first. The response returned will be 401 Unauthorized since there is no token included in the request header for the Get Food API call.

Step 8: Next, we can run the “Get Food (403 Forbidden)”. The response returned will be 403 Forbidden because the token is unrecognised even though there is a token included in the request header for the Get Food API call.

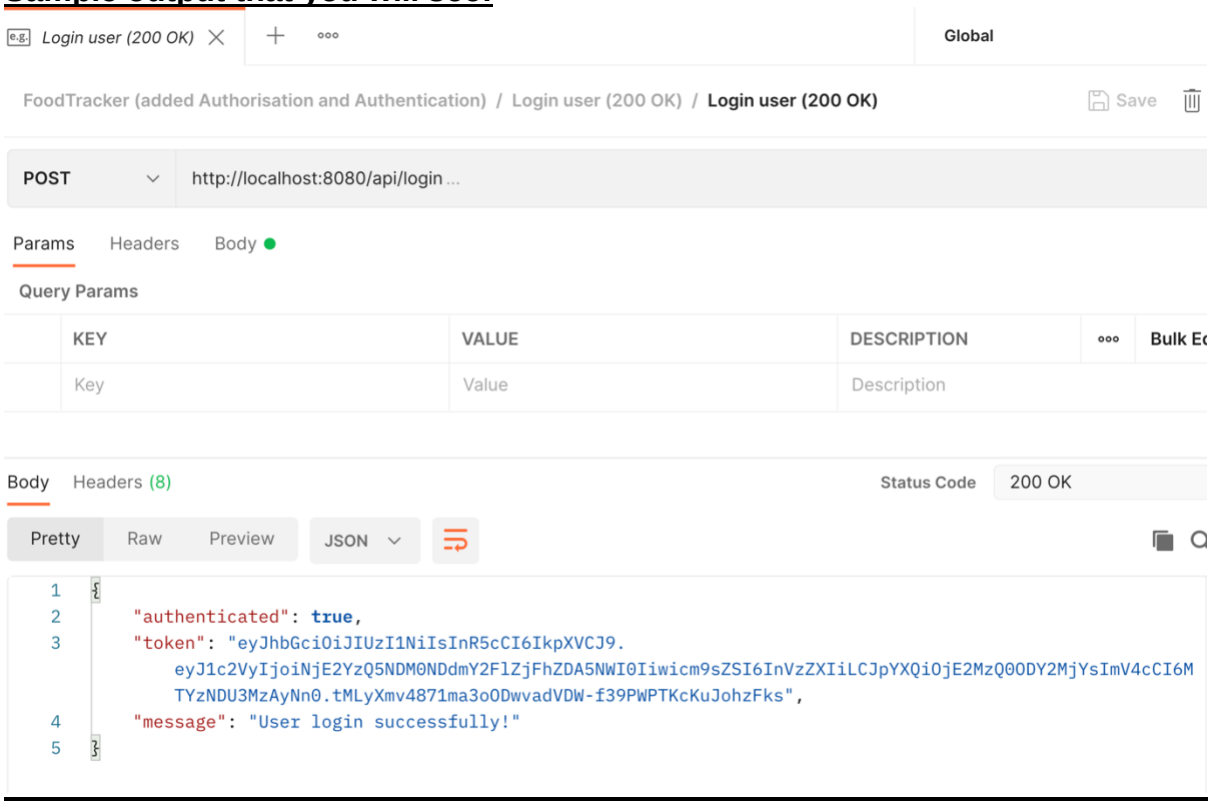


Step 9: We can run “Register user” and “Register admin” API calls to have a normal user account and an admin user.

Step 10: We can run the “Invalid login (401 Unauthorized)” API call. The response returned is 401 Unauthorized where the user has entered a wrong password.

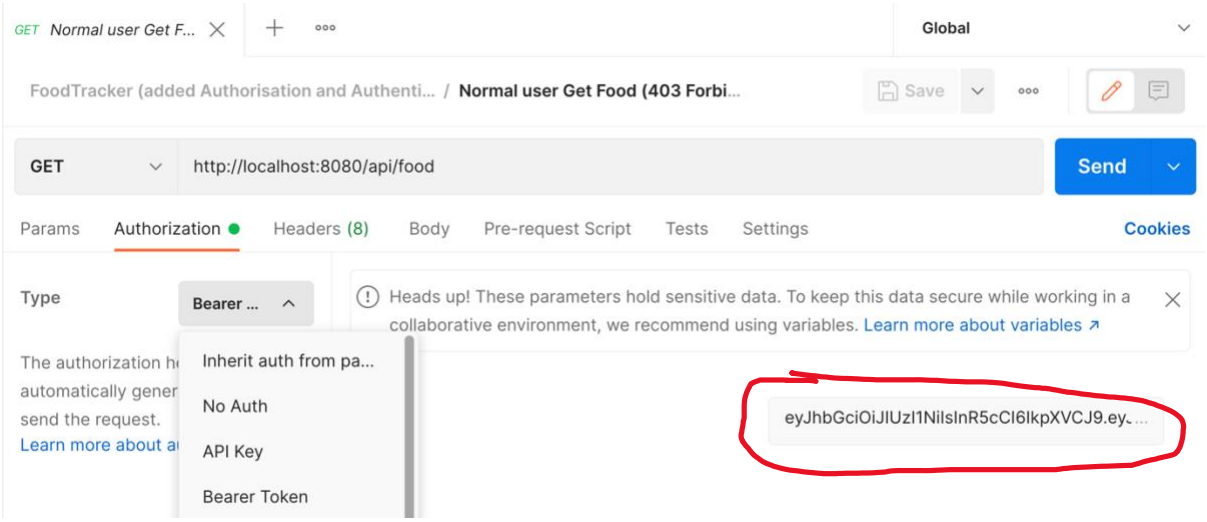
Step 11: We can then run “Login user (200 OK)” and “Login admin (200 OK)” API calls where the credentials of both admin and user matches what they have entered during the “Register user” and “Register admin” API calls respectively.

Sample output that you will see:

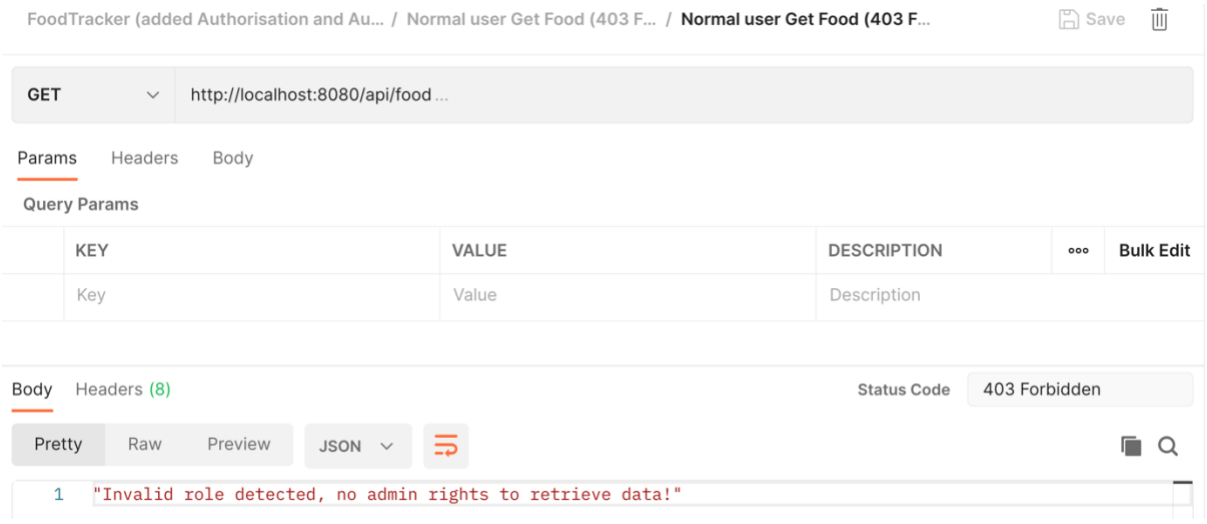


At this stage, please copy down the token for both user and admin somewhere. This is because every time you login a new jwt token will be signed and the token will last a day only.

Step 12: Next, we can perform “Normal user Get Food (403 Forbidden)” API call where you can paste in the user token from Step 10 in the bearer token option under the authorization tab as shown below.



You will receive a 403 Forbidden response as shown below because only admin has access to all the food data.



Step 13: Now, if you remain on the same Postman tab, you can just replace the token with the admin token from step 10 and run the API call again. This time you should see a 200 OK response because the token that you passed in has been verified that you are an admin and have sufficient privileges to access the food data.

References

1) https://stackabuse.com/authentication-and-authorization-with-jwts-in-express-js/
2) https://dev.to/aditya278/understanding-and-implementing-password-hashing-in-nodejs-2m84