Employee Management System

Please check the the project setup guide in the readme.md from Github (https://github.com/Uncaught-TypeError/employee-management-app) before start testing the system.

Introduction

Web Backend Developer Trainee Assignment - Employee Management API

Submission Deadline - (48hrs)

This assignment aims to develop a RESTful API for managing employee data using Laravel. It is part of my evaluation process for web backend developer trainees position at Better HR. The overall evaluation process includes implementing user registration, authentication, employee creation, listing, update, and deletion and database management.

Scope & Requirements

- User registration and authentication using Laravel Passport
- CRUD Operation for Employees (Create, Read, Update, Delete)
- Validation and error handling for API requests

Testing

In this assignment, i used light REST API client, Thunder Client for testing APIs.

Install Thunder Client Extension in VS Code.

User Registeration

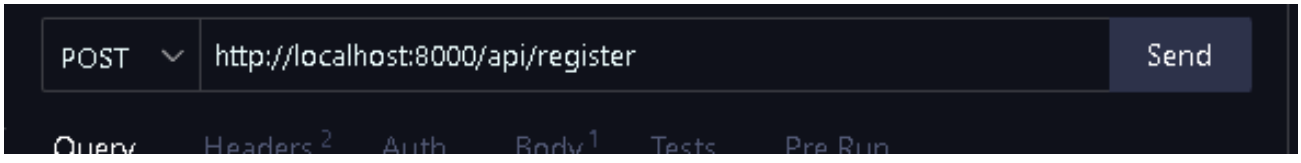
#Step 1: Open Thunder Client Extension.

#Step 2: Create New Request.

- Click new request button.

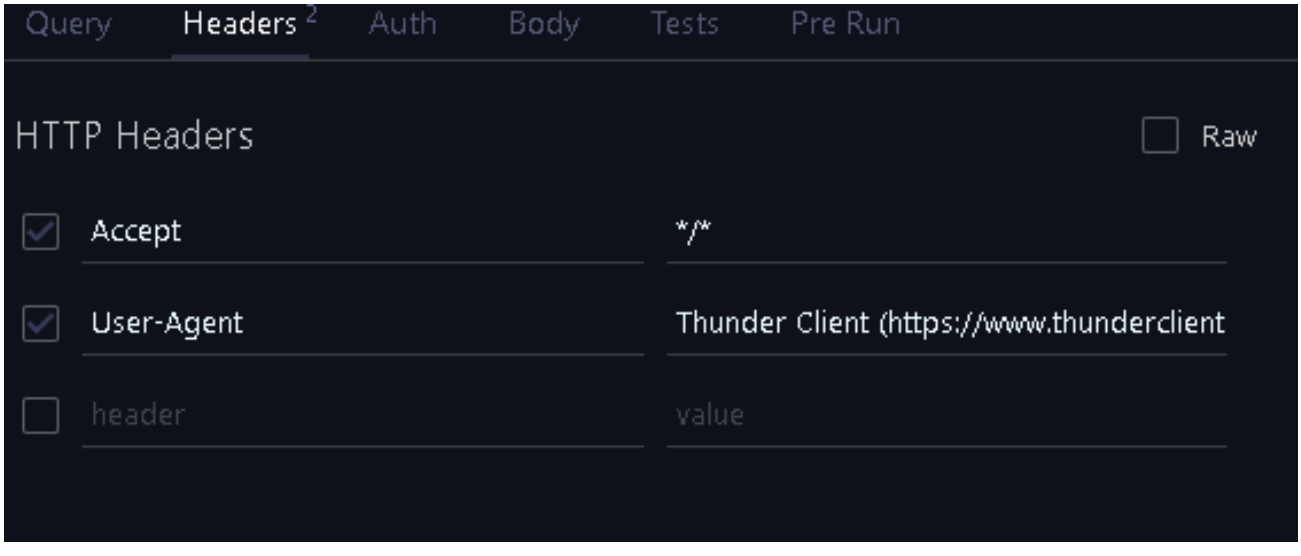
#Step 3: Change the method and input Request URL.

- Click the dropdown button and select POST.
- Inside the request URL bar, type in “http://localhost:8000/api/register”.



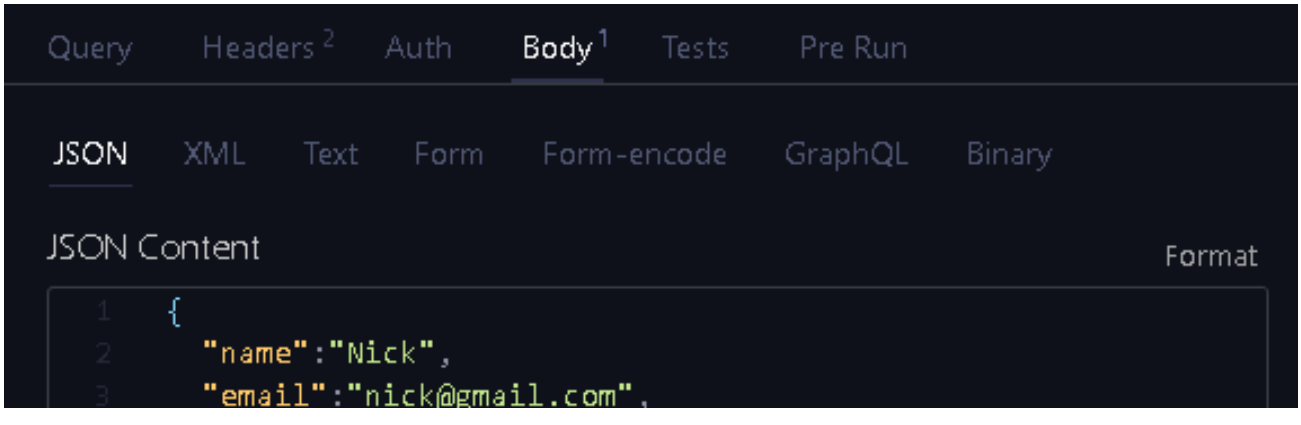
#Step 4: Go to Headers.

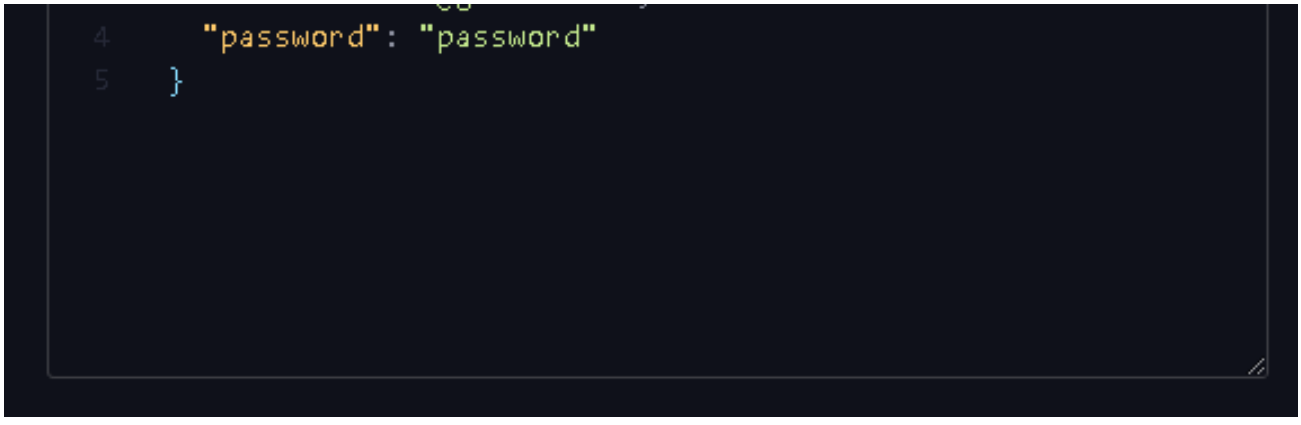
- Go to Headers and remove “*/*” and type in “application/json”.



#Step 5: Go to Body.

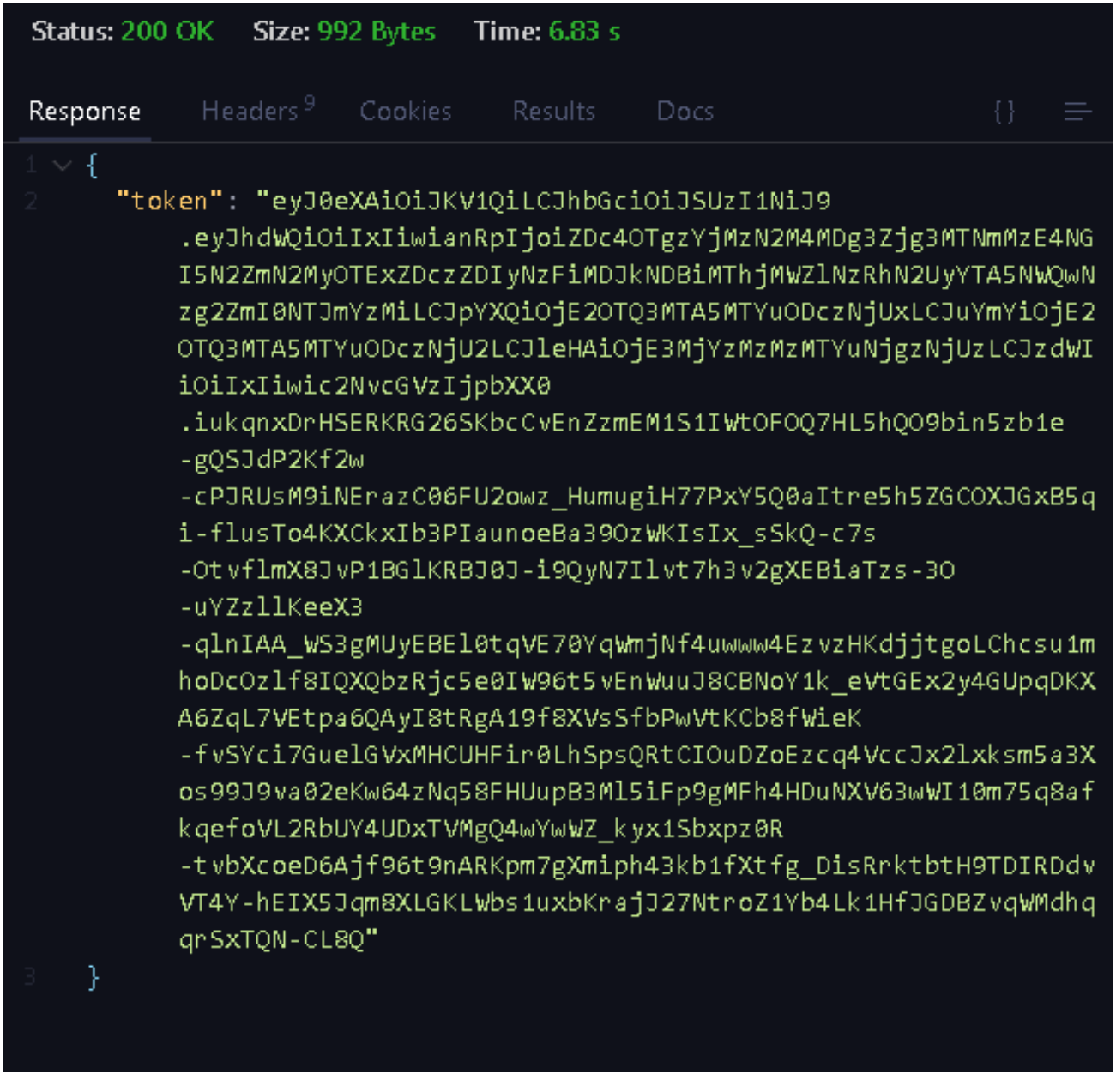
- Go to Body and type in JSON Content.
- In this case, user’s name, email and password.





#Step 6: Click Send.

- If the registeration is successful, you will see status 200 along with a token which you need to use for the rest of the operation.



- If the registeration has failed due to the validation, you will see status 422 along with a message for the error (in this case, email field).

Employee CRUD Operation

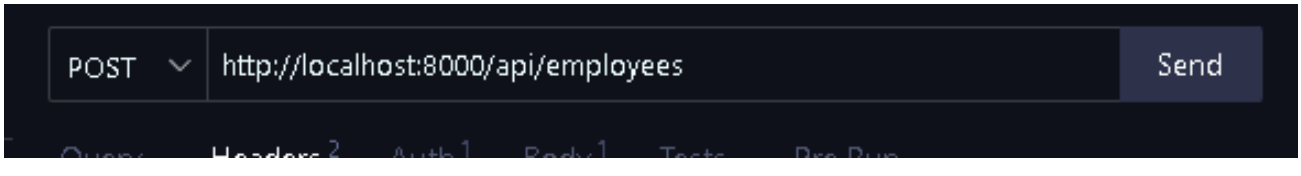
Creating Employee

#Step 1: Create New Request.

- Click new request button.

#Step 2: Change the method and input Request URL.

- Click the dropdown button and select POST.
- Inside the request URL bar, type in “http://localhost:8000/api/employees”.

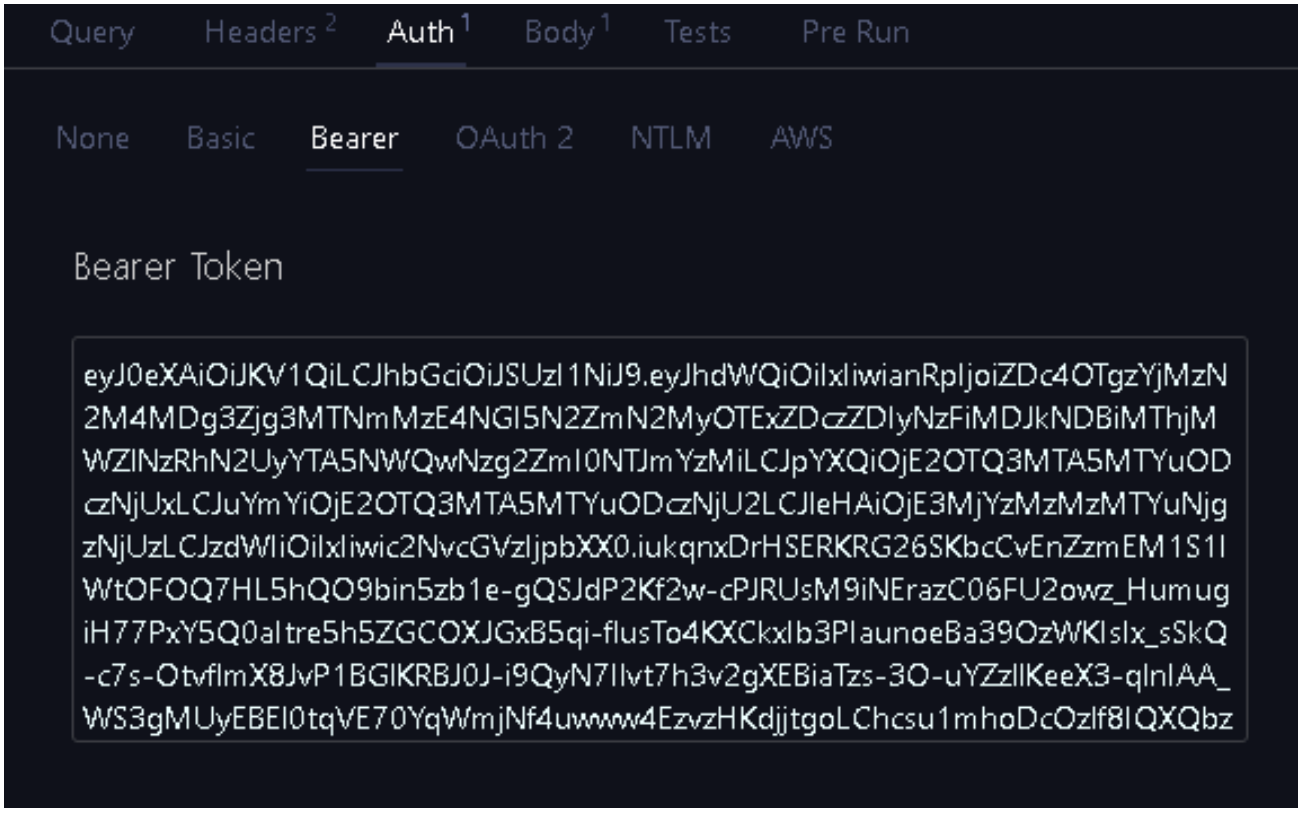


#Step 4: Go to Headers.

- Go to Headers and remove “*/*” and type in “application/json”.

#Step 5: Go to Auth.

- Go to Auth → Bearer → Inside Bearer Token, Paste the Token from the previous registeration.

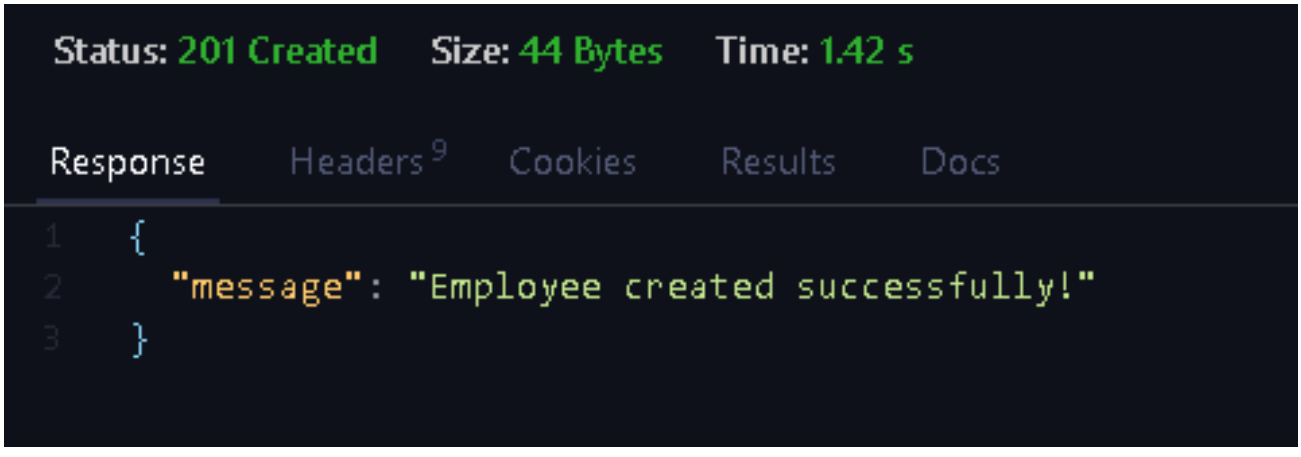


#Step 6: Go to Body.

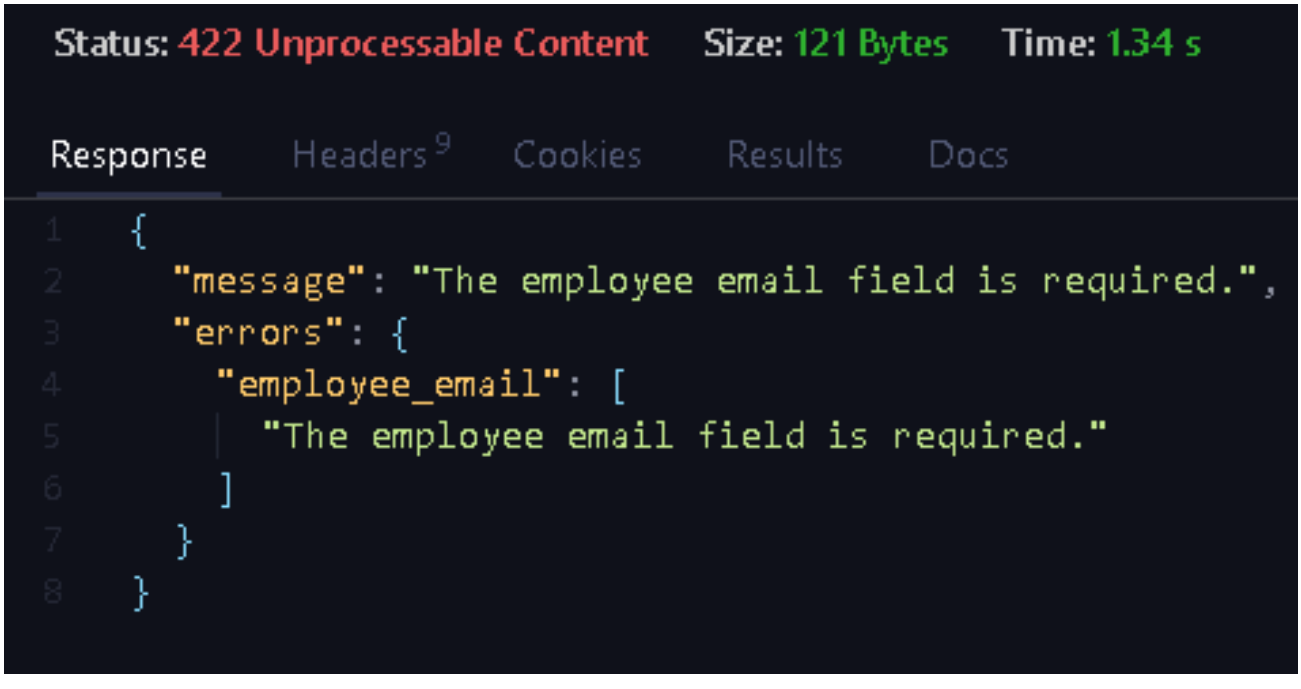
- Go to Body and type in JSON Content.
- In this case, employee's name, email and age.

#Step 7: Click Send.

- If the registeration is successful, you will see status 201 along with a success message.



- If the registeration has failed, you will see status 422 along with a error message.



Reading Employees and Checking Employee List

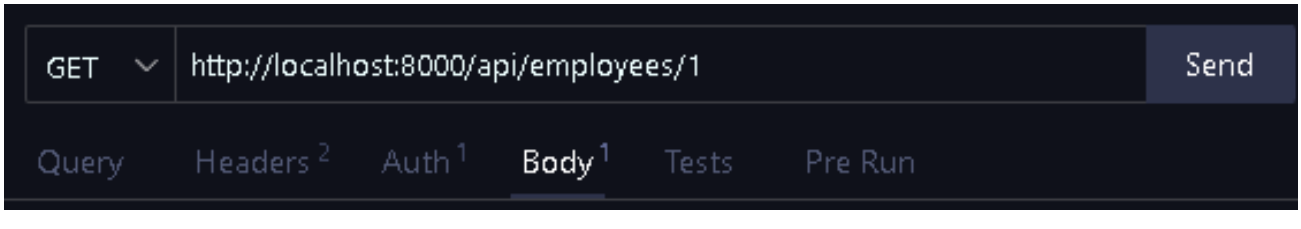
#Step 1: Continuing from the step above.

#Step 2: Change the method and input Request URL.

- Click the dropdown button and select GET.
- Inside the request URL bar, type in “http://localhost:8000/api/employees” and click send.
- You will see all the employees created by the user(the token’s user).

#Step 3: Change Request URL.

- Inside the request URL bar, type in “http://localhost:8000/api/employees/[employee_id]”.



- You can see specific employee based on their IDs.

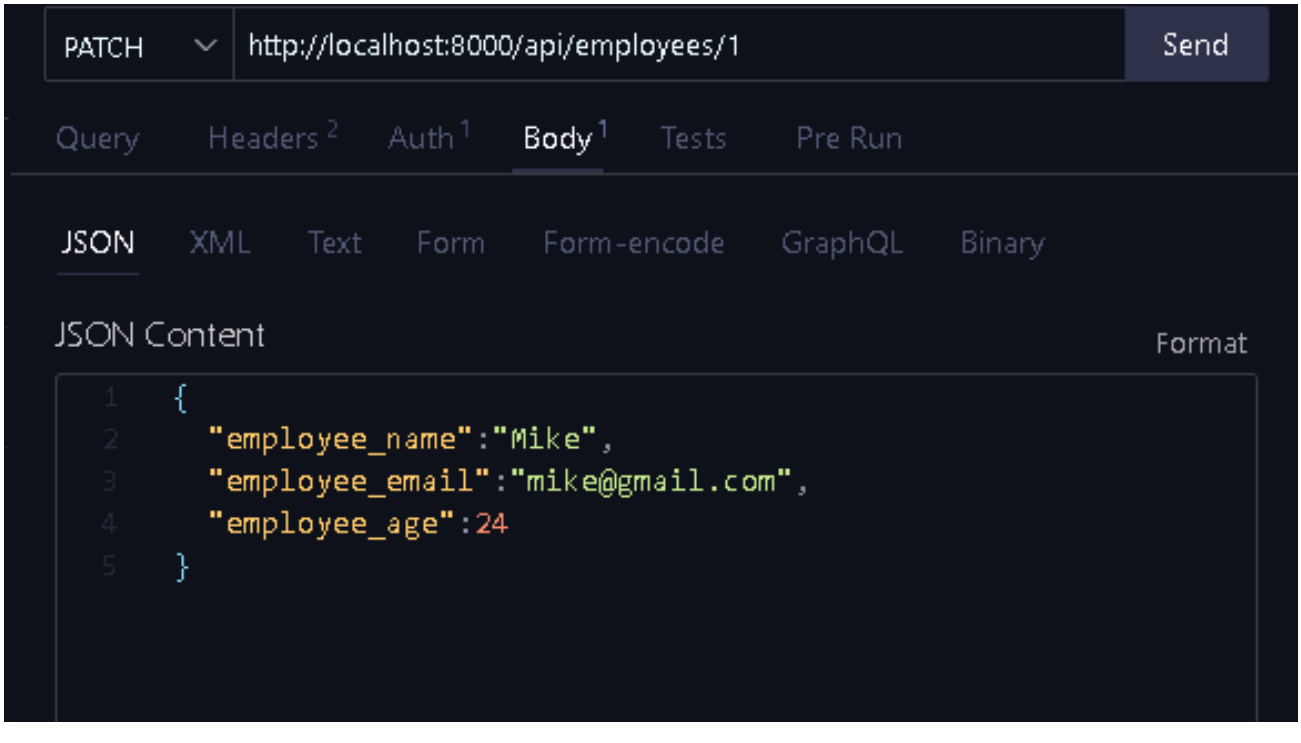
```
{
  "id": 1,
  "employee_name": "Mick",
  "employee_email": "mick@gmail.com",
  "employee_age": 45,
  "user_id": 1,
  "created_at": "2023-09-14T17:04:08.000000Z",
  "updated_at": "2023-09-14T17:04:08.000000Z"
}
```


Updating Employees

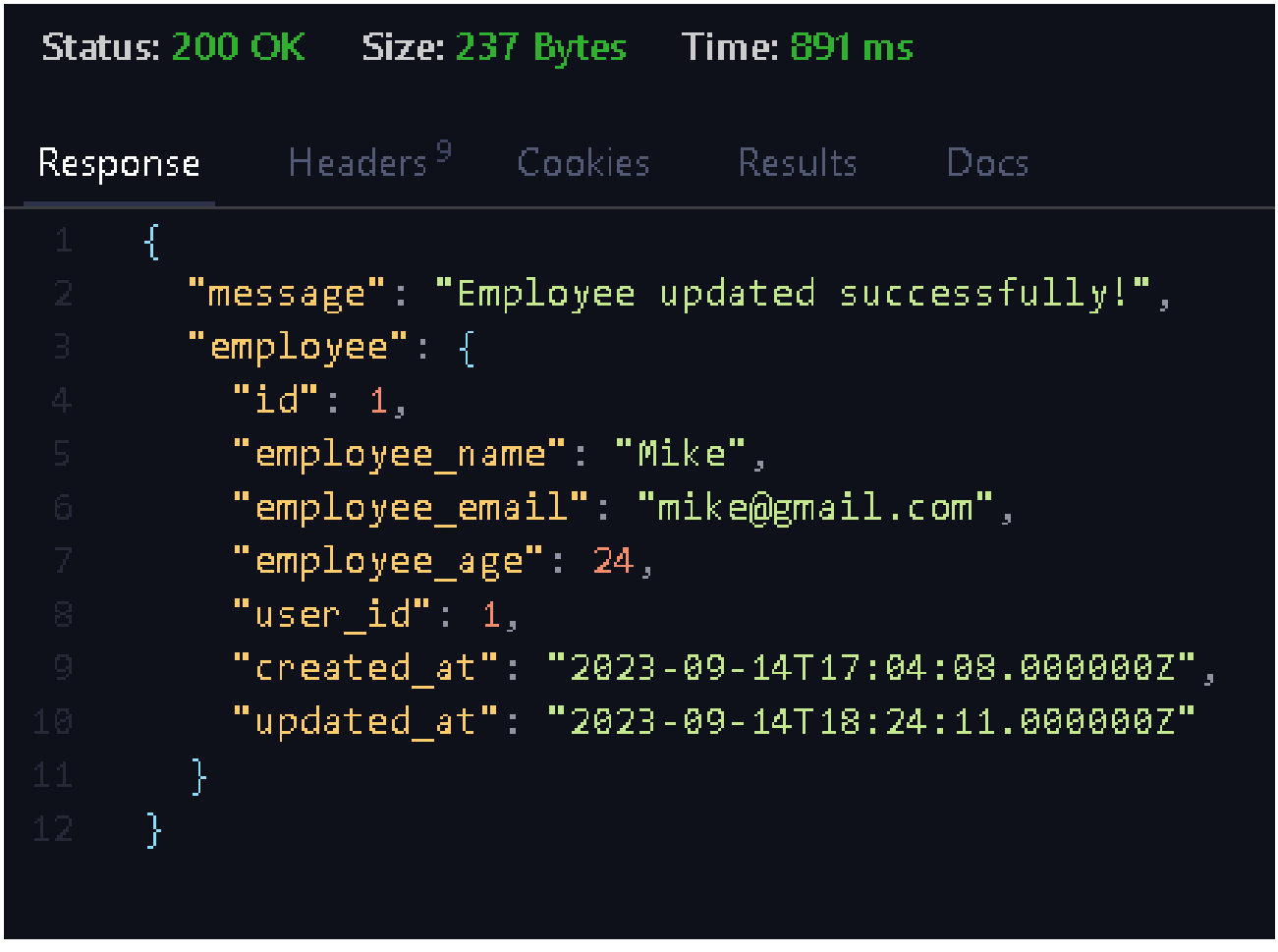
#Step 1: Continuing from the step above.

#Step 2: Change the method and input Request URL.

- Click the dropdown button and select PATCH.
- Inside the request URL bar, type in “http://localhost:8000/api/employees/[employee_id]”.
- Inside JSON Content, type in employee’s data that want to be changed (In this case, from Mick to Mike) and click send.



- If successful, you can see status message with 200 and the updated data together with the success message. If not, there will be an error message just like above.



Deleting Employees

#Step 1: Continuing from the step above.

#Step 2: Change the method and input Request URL.

- Click the dropdown button and select DELETE.
- Inside the request URL bar, type in “http://localhost:8000/api/employees/[employee_id]” and click send.

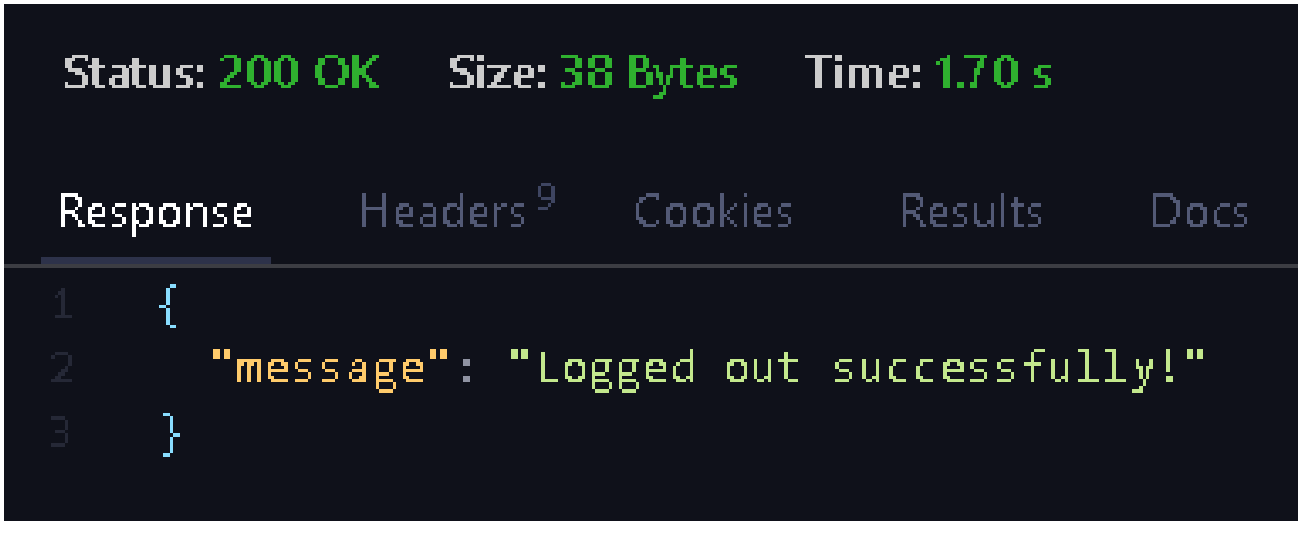
- If the delete is successful, you can see a status message with 200.

User Logout

#Step 1: Continuing from the step above.

#Step 2: Change the method and input Request URL.

- Click the dropdown button and select POST.
- Inside the request URL bar, type in “http://localhost:8000/api/logout” and click send.



This includes the testing of CRUD operation of the employees and user registeration and login.

Conclusion

In conclusion, this task has allowed me to gain hands-on experience in developing a RESTful API for employee management system using the laravel framework. Aside from some small obstacles that i faced, i managed to successfully implemented key features which are user registeration, authentication via Laravel Passport and Employee Management (CRUD).

To be honest, Laravel passport is a package which was a bit quite unfamiliar to me but with the help of documentations and tutorials, i managed to successfully made it. However, there is always room for improvement and in terms of meeting the assignment objectives, i believe i have made significant progress in mastering Laravel-based API developments. And i have no doubt believes future work could involve more refining API endpoints, role based access controls such as Spatie, error handling and so on.

In closing, I welcome any feedback or suggestions for improvement and am eager to continue improving my skills as a web backend developer.