

# 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 Registration

**#Step 1: Open Thunder Client Extension.**

**#Step 2: Create New Request.**

- Click new request button.



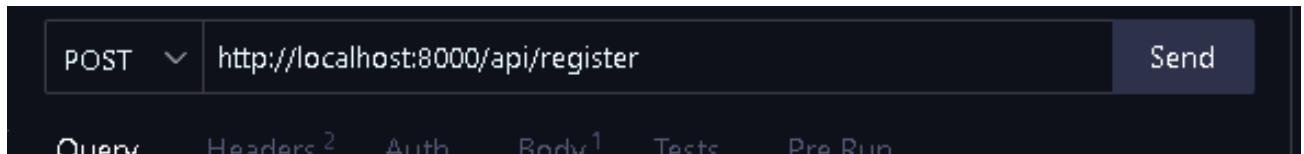
Activity

Collections

Env

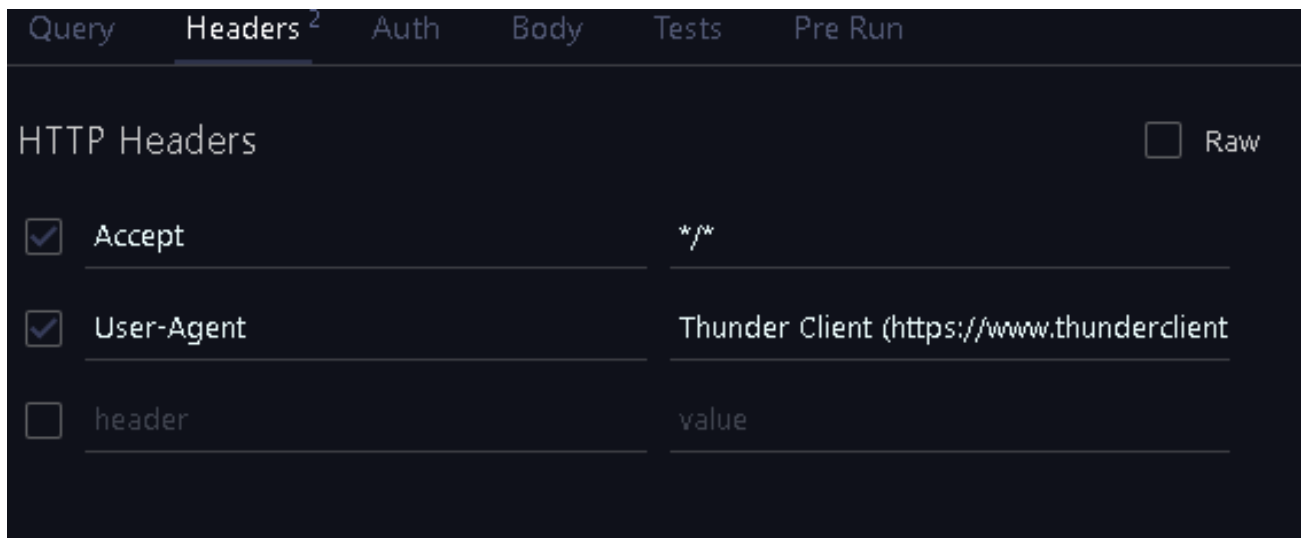
### #Step 3: Change the method and input Request URL.

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



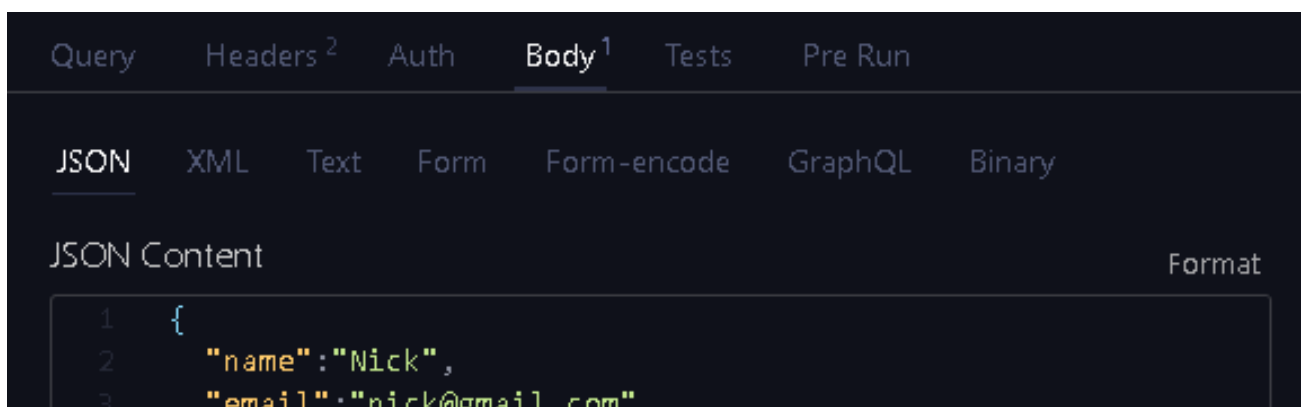
### #Step 4: Go to Headers.

- 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.



```

4     "password": "password"
5 }

```

## #Step 6: Click Send.

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

Status: 200 OK Size: 992 Bytes Time: 6.83 s

Response

Headers<sup>9</sup>

Cookies

Results

Docs

{}

1 {

```

2     "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9
      .eyJhdWQiOiIiXiwianRpIjoiZDc4OTgzYjMzN2M4MDg3Zjg3MTNmMzE4NG
      I5N2ZmN2MyOTExZDczZDIyNzFiMDJkNDBiMThjMwZlNzRhN2UyYTA5NWQwN
      zg2ZmI0NTJmYzMiLCJpYXQiOiJE2OTQ3MTA5MTYuODczNjUxLCJyYmYiOiJE2
      OTQ3MTA5MTYuODczNjU2LCJleHAiOiJE3MjYzMzM5MTYuNjgzNjUzLCJzZW
      iOiIiXiwic2NvcGVzIjpbXX0
      .iukqnxDrHSERKRG26SKbcCvEnZzmEM1S1IWtOFOQ7HL5hQO9bin5zb1e
      -gQSJdP2Kf2w
      -cPJRUsM9iNErazC06FU2owz_HumugiH77PxY5Q0aItre5h5ZGCOXJGxB5q
      i-flusTo4KXCkxIb3PIaunoeBa390zWKIsIx_sSkQ-c7s
      -OtvflmX8JvP1BG1KRBJ0J-i9QyN7I1vt7h3v2gXEBiaTzs-30
      -uYZz11KeeX3
      -qInIAA_WS3gMUyEBEl0tqVE70YqWmjNf4uww4Ez vzHKdjtgolChcsu1m
      hoDcOz1f8IQXQbzRjc5e0IW96t5vEnWuuJ8CBNoY1k_eVtGEx2y4GUpqDKX
      A6ZqL7VEtpa6QAyI8tRgA19f8XVsSfbPwVtKCb8fwiek
      -fvSYci7GuelGVxMHCUHFir0LhSpsQRtCIouDZoEzcq4VccJx2lXksm5a3X
      os99J9va02eKw64zNq58FHUupB3M15iFp9gMFh4HDuNXV63wWI10m75q8af
      kqefoVL2RBUY4UDxTVMgQ4wYwWZ_kyx1SbXPz0R
      -tVbXcoeD6Ajf96t9nARKpm7gXmiph43kb1fXtfg_DisRrktbtH9TDIRDdv
      VT4Y-hEIX5Jqm8XLGKLWbs1uxbKraJJ27NtroZ1Yb4Lk1HfJGDBZvqWmdhq
      qrSxTQN-CL8Q"

```

3 }

- If the registration 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

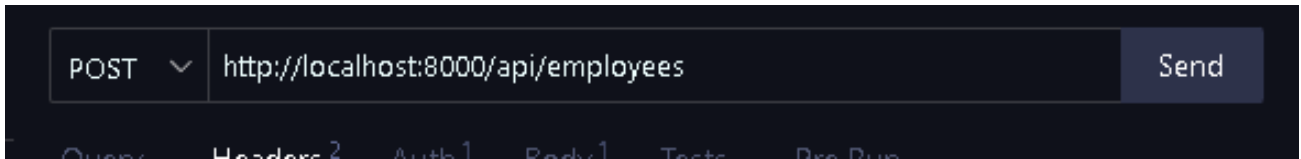
### Creating Employee

#### #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.

- Query Headers <sup>2</sup> Auth <sup>1</sup> Body <sup>1</sup> Tests Pre Run
- None Basic Bearer OAuth 2 NTLM AWS
- Bearer Token
- ```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiJxliwianRpljoiZDc4OTgzYjMzM2M4MDg3Zjg3MTNmMzE4NGI5N2ZmN2MyOTExZDczZDliNzFjMDJkNDBiMThjMwZlInZRhN2UyYTA5NWQwNzg2ZmI0NTJmYzMiLCJpYXQiOiJlY2OTQ3MTA5MTYuODczNjUxLCJyYmYiOiJlY2OTQ3MTA5MTYuODczNjU2LCJleHAiOiJlY3MjYzZmZmMTYuNjgzNjUzLCJzdWliOiJxliwic2NvcGVzIjpbXX0.iukqnxDrHSERKRG26SKbcCvEnZzmEM1S1lWtOFOQ7HL5hQO9bin5zb1e-gQSJdP2Kf2w-cPJRUsM9iNErazC06FU2owz_Humugih77PxY5Q0altre5h5ZGCOXJGxB5qi-flusTo4KXCkxlb3PlaunoeBa39OzWKLslx_sSkQ-c7s-OtvflmX8JvP1BGIKRBJ0J-i9QyN7lIvt7h3v2gXEBiaTzs-3O-uYZZllKeeX3-qlnlAA_WS3gMUyEBEI0tqVE70YqWmjNf4uwww4EzvvHKdjtgolChcsu1mhoDcOzlf8lQXQbz
```

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

A screenshot of the REST Client application. The top bar contains tabs for 'Query', 'Headers', 'Auth', 'Body', 'Tests', and 'Pre Run'. The 'Body' tab is selected and highlighted. Below the tabs, there are sub-tabs for 'JSON', 'XML', 'Text', 'Form', 'Form-encode', 'GraphQL', and 'Binary'. The 'JSON' sub-tab is selected. The main area is titled 'JSON Content' and shows a JSON object with three properties: 'employee\_name' with value 'Mick', 'employee\_email' with value 'mick@gmail.com', and 'employee\_age' with value 45. The JSON is displayed with syntax highlighting and line numbers (1-5) on the left. A 'Format' button is visible in the top right corner of the JSON content area.

## #Step 7: Click Send.

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

Status: 201 Created    Size: 44 Bytes    Time: 1.42 s

Response    Headers<sup>9</sup>    Cookies    Results    Docs

1    {  
2        "message": "Employee created successfully!"  
3    }

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

Status: 422 Unprocessable Content    Size: 121 Bytes    Time: 1.34 s

Response    Headers<sup>9</sup>    Cookies    Results    Docs

1    {  
2        "message": "The employee email field is required.",  
3        "errors": {  
4            "employee\_email": [  
5                "The employee email field is required."  
6            ]  
7        }  
8    }

## 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).

Status: 200 OK   Size: 365 Bytes   Time: 1.13 s

Response

Headers<sup>9</sup>

Cookies

Results

Docs

```
1  [
2    {
3      "id": 1,
4      "employee_name": "Mick",
5      "employee_email": "mick@gmail.com",
6      "employee_age": 45,
7      "user_id": 1,
8      "created_at": "2023-09-14T17:04:08.000000Z",
9      "updated_at": "2023-09-14T17:04:08.000000Z"
10   },
11   {
12     "id": 2,
13     "employee_name": "Hiro",
14     "employee_email": "hiro@gmail.com",
15     "employee_age": 24,
16     "user_id": 1,
17     "created_at": "2023-09-14T18:15:44.000000Z",
18     "updated_at": "2023-09-14T18:15:44.000000Z"
19   }
20 ]
```

### #Step 3: Change Request URL.

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

GET



http://localhost:8000/api/employees/1

Send

Query

Headers<sup>2</sup>

Auth<sup>1</sup>

Body<sup>1</sup>

Tests

Pre Run

- You can see specific employee based on their IDs.

Status: 200 OK   Size: 181 Bytes   Time: 748 ms

Response

Headers<sup>9</sup>

Cookies

Results

Docs

```
1  {
2    "id": 1,
3    "employee_name": "Mick",
4    "employee_email": "mick@gmail.com",
5    "employee_age": 45,
6    "user_id": 1,
7    "created_at": "2023-09-14T17:04:08.000000Z",
8    "updated_at": "2023-09-14T17:04:08.000000Z"
9  }
```

## 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.

The screenshot shows a REST client interface with the following components:

- Method:** PATCH (selected from a dropdown)
- URL:** http://localhost:8000/api/employees/1
- Buttons:** Send
- Tabs:** Query, Headers<sup>2</sup>, Auth<sup>1</sup>, Body<sup>1</sup> (selected), Tests, Pre Run
- Body Format:** JSON (selected from a dropdown), XML, Text, Form, Form-encode, GraphQL, Binary
- JSON Content:**

```
1  {
2    "employee_name": "Mike",
3    "employee_email": "mike@gmail.com",
4    "employee_age": 24
5  }
```
- Format:** A button to format the JSON content.



- 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.

Status: **200 OK**   Size: **237 Bytes**   Time: **891 ms**

Response

Headers<sup>9</sup>

Cookies

Results

Docs

```
1  {
2    "message": "Employee updated successfully!",
3    "employee": {
4      "id": 1,
5      "employee_name": "Mike",
6      "employee_email": "mike@gmail.com",
7      "employee_age": 24,
8      "user_id": 1,
9      "created_at": "2023-09-14T17:04:08.000000Z",
10     "updated_at": "2023-09-14T18:24:11.000000Z"
11   }
12 }
```

## 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.

DELETE



http://localhost:8000/api/employees/1

Send

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

Status: **200 OK**   Size: **44 Bytes**   Time: **836 ms**

| Response | Headers <sup>9</sup>                        | Cookies | Results | Docs |
|----------|---------------------------------------------|---------|---------|------|
| 1        | {                                           |         |         |      |
| 2        | "message": "Employee deleted successfully!" |         |         |      |
| 3        | }                                           |         |         |      |

## 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.

Status: 200 OK    Size: 38 Bytes    Time: 1.70 s

| Response | Headers <sup>9</sup>                  | Cookies | Results | Docs |
|----------|---------------------------------------|---------|---------|------|
| 1        | {                                     |         |         |      |
| 2        | "message": "Logged out successfully!" |         |         |      |
| 3        | }                                     |         |         |      |

This includes the testing of CRUD operation of the employees and user registration 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 registration, 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.