Contents

[1.Your application should be able to add roles in the database dynamically in the db. 1](#_Toc109121328)

[2. Your application should be able to add Users in the db which can be used for authentication purposes. 3](#_Toc109121329)

[3. Now Your application should be able to add employees data in the db if and 5](#_Toc109121330)

[4. Your application should provide an endpoint to list all the employees stored in the database 6](#_Toc109121331)

[5. Your application should provide endpoint to fetch or get an employee record specifically based on the id of that employee- 7](#_Toc109121332)

[6. Your application should provide an endpoint to update an existing employee record with the given updated json object. 8](#_Toc109121333)

[7. Your application should also provide an endpoint to delete an existing employee record based on the id of the employee- 9](#_Toc109121334)

[8. Your application should provide an endpoint to fetch an employee by his/her first name and if found more than one record then list them all- 10](#_Toc109121335)

[9. Your application should be able to list all employee records sorted on their first name in either ascending order or descending order . 11](#_Toc109121336)

**Question-**

You are required to create a Employee Management Rest Api based Web application, where you will be developing CRUD(Create,Read,Update and Delete) functionality along with Sorting and some concepts of security.

Your Rest Api should be secure.And should have different endpoints for different operations-

|  |  |
| --- | --- |
| **Endpoint** | **Roles required** |
| /roles/create | No Roles are required |
| /users/register | No Roles are required |
| /api/employees/save | Admin |
| /api/employees/getAllEmployees | Admin,user |
| /api/employees/getEmployeeById/\*\* | Admin,user |
| /api/employees/search/\*\* | Admin,user |
| /api/employees/sort/\*\* | Admin,user |
| /api/employees/update | Admin |
| /api/employees/delete/\*\* | Admin |

## 1.Your application should be able to add roles in the database dynamically in the db.

Request URL: localhost:8080/roles/create

Ex payload-

{

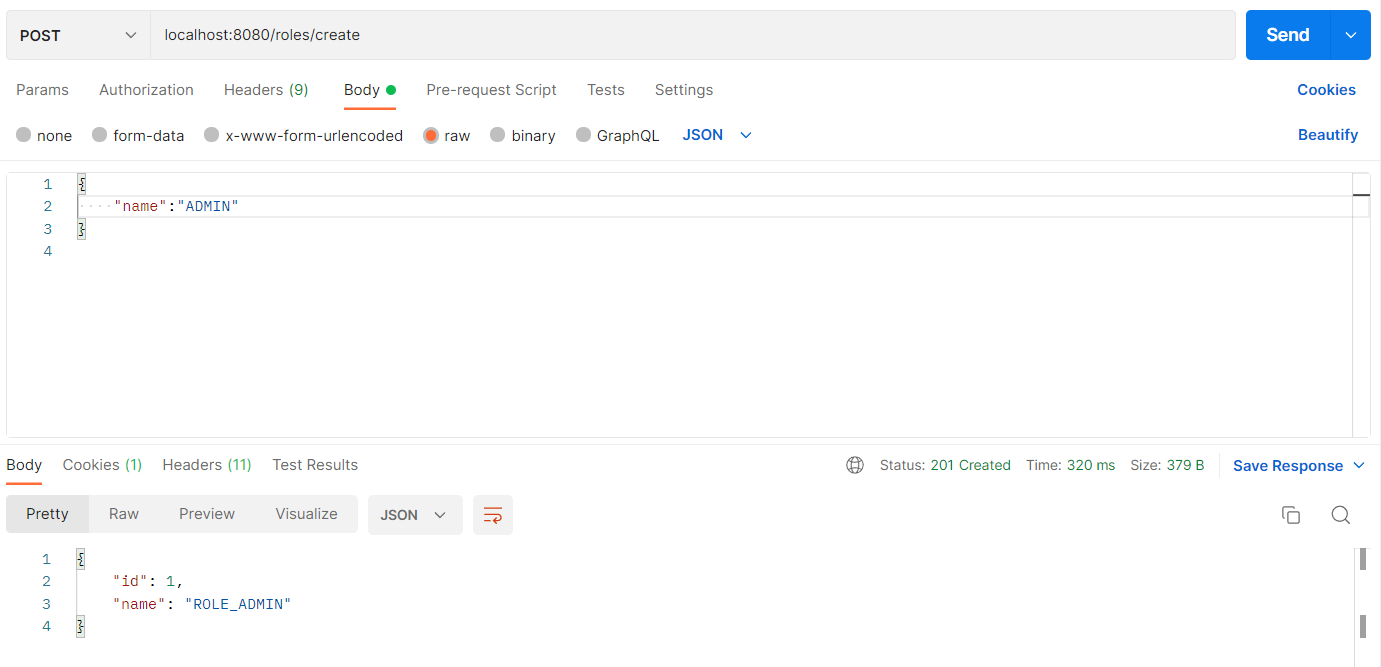
"name":"USER"

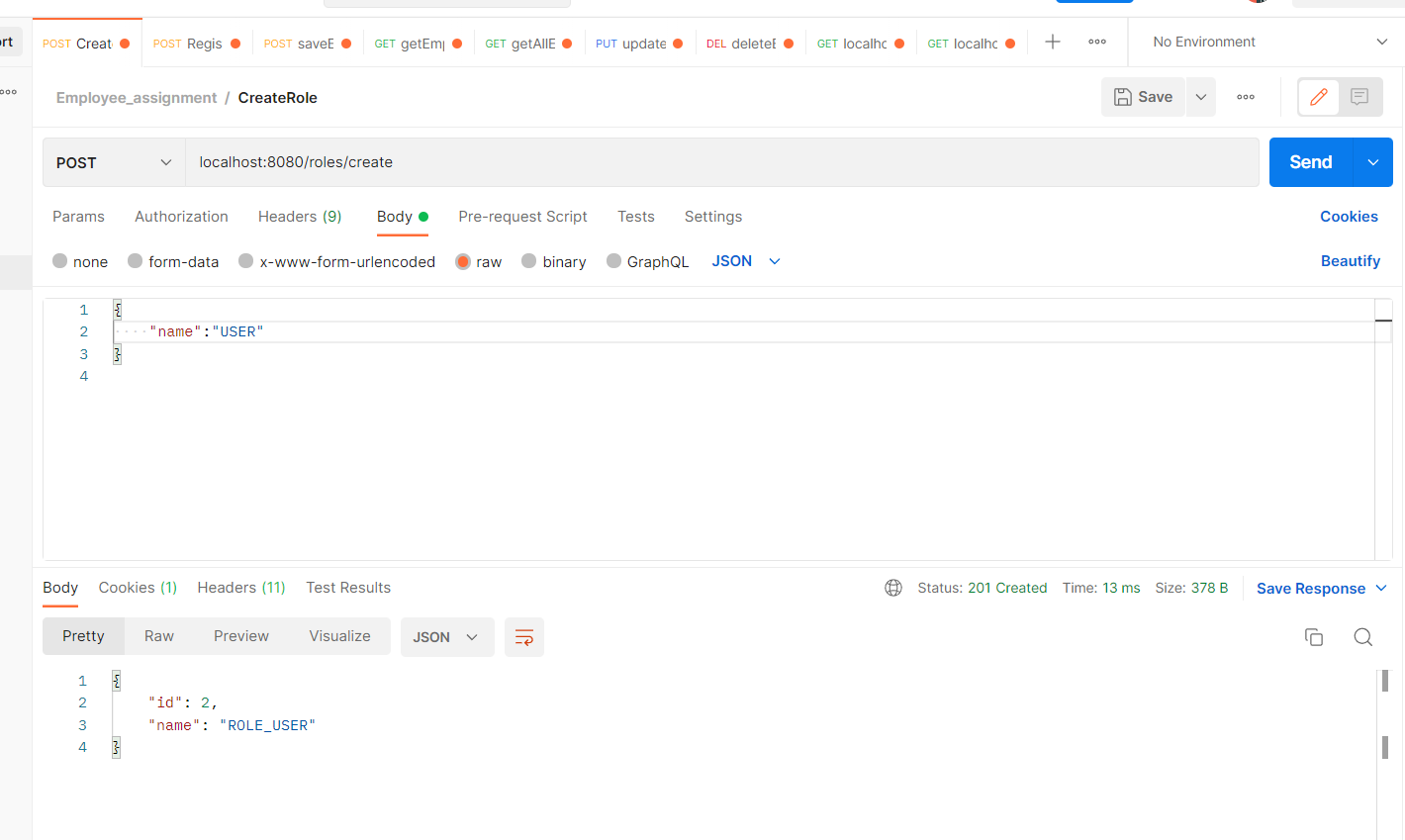
}

{

    "name":"ADMIN"

}





# 2. Your application should be able to add Users in the db which can be used for authentication purposes.

**Creating user with Admin Role:**

localhost:8080/users/register

{

    "username":"admin",

    "password":"admin",

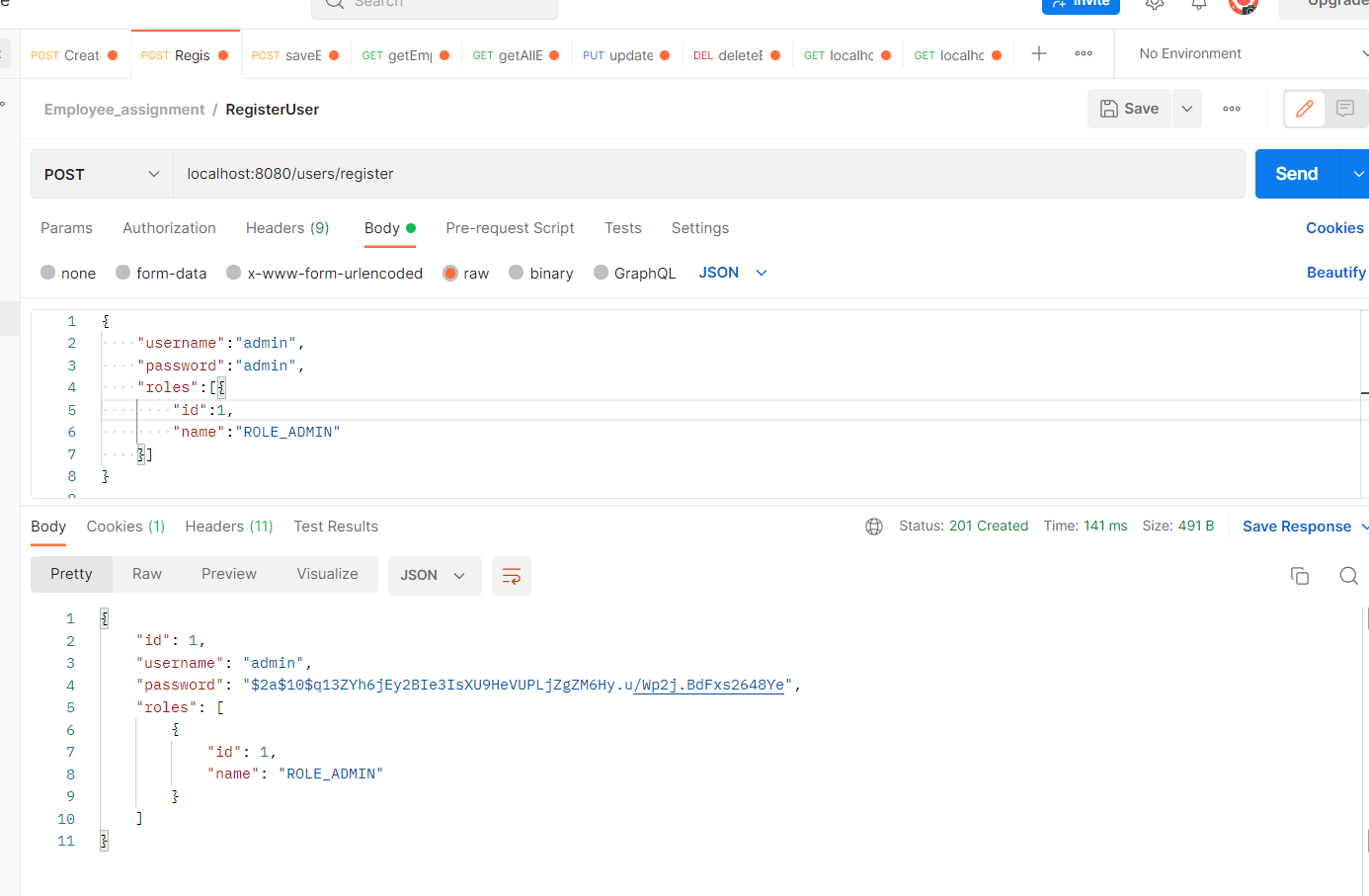
    "roles":[{

        "id":1,

        "name":"ROLE\_ADMIN"

    }]

}



Creating user with user role:

localhost:8080/users/register

{

    "username":"john",

    "password":"john",

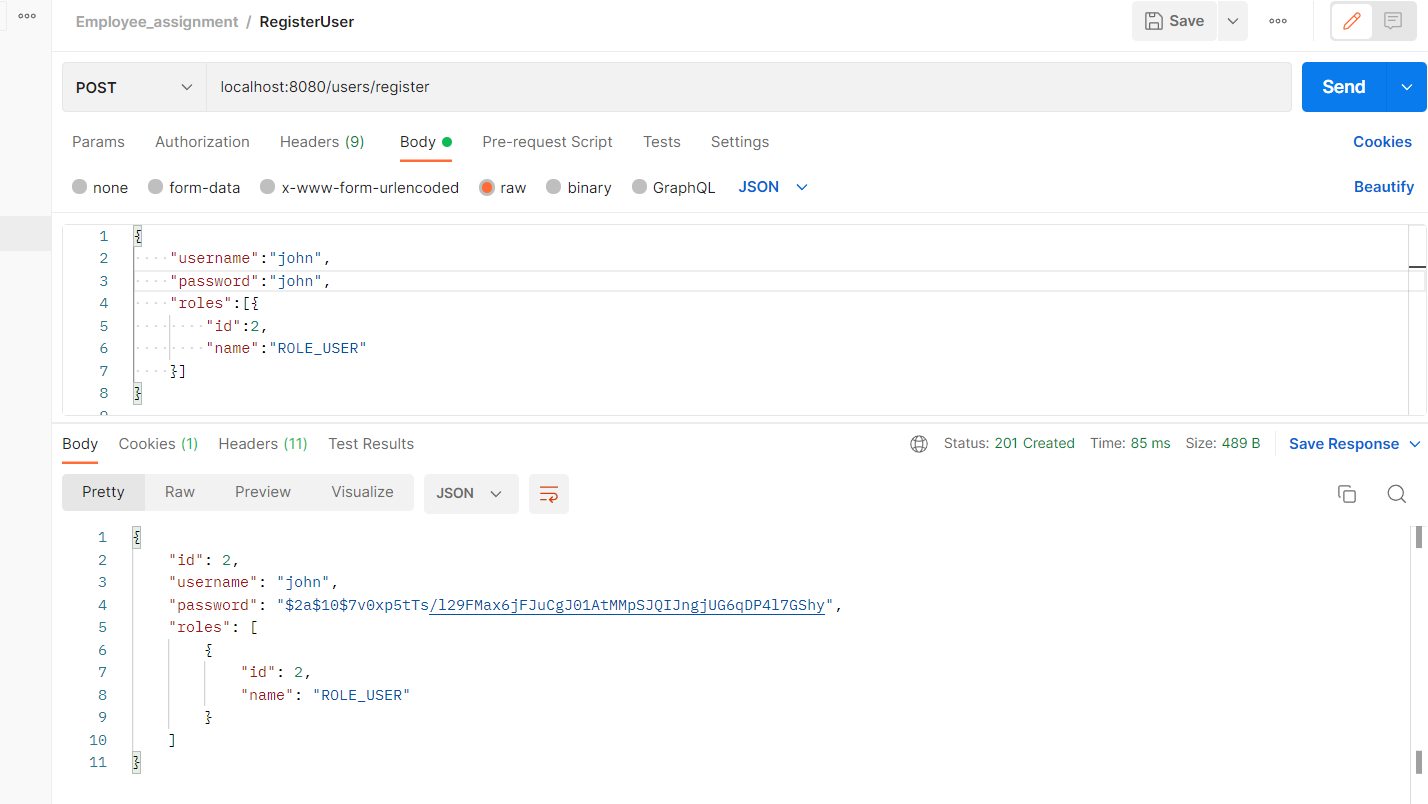
    "roles":[{

        "id":2,

        "name":"ROLE\_USER"

    }]

}



3. Now Your application should be able to add employees data in the db if and only if the authenticated user is **ADMIN**-

Ex-

{

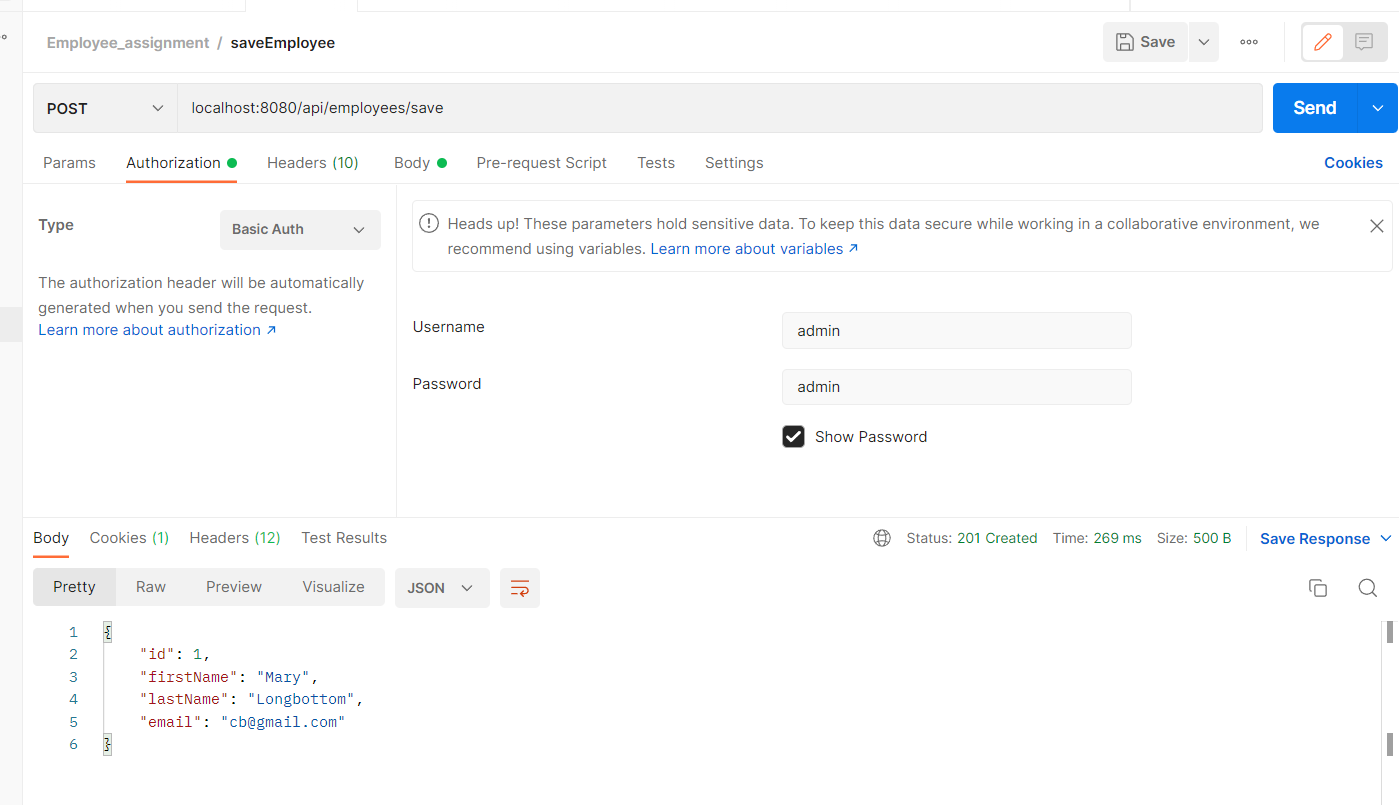
    "firstName": "Mary",

    "lastName": "Longbottom",

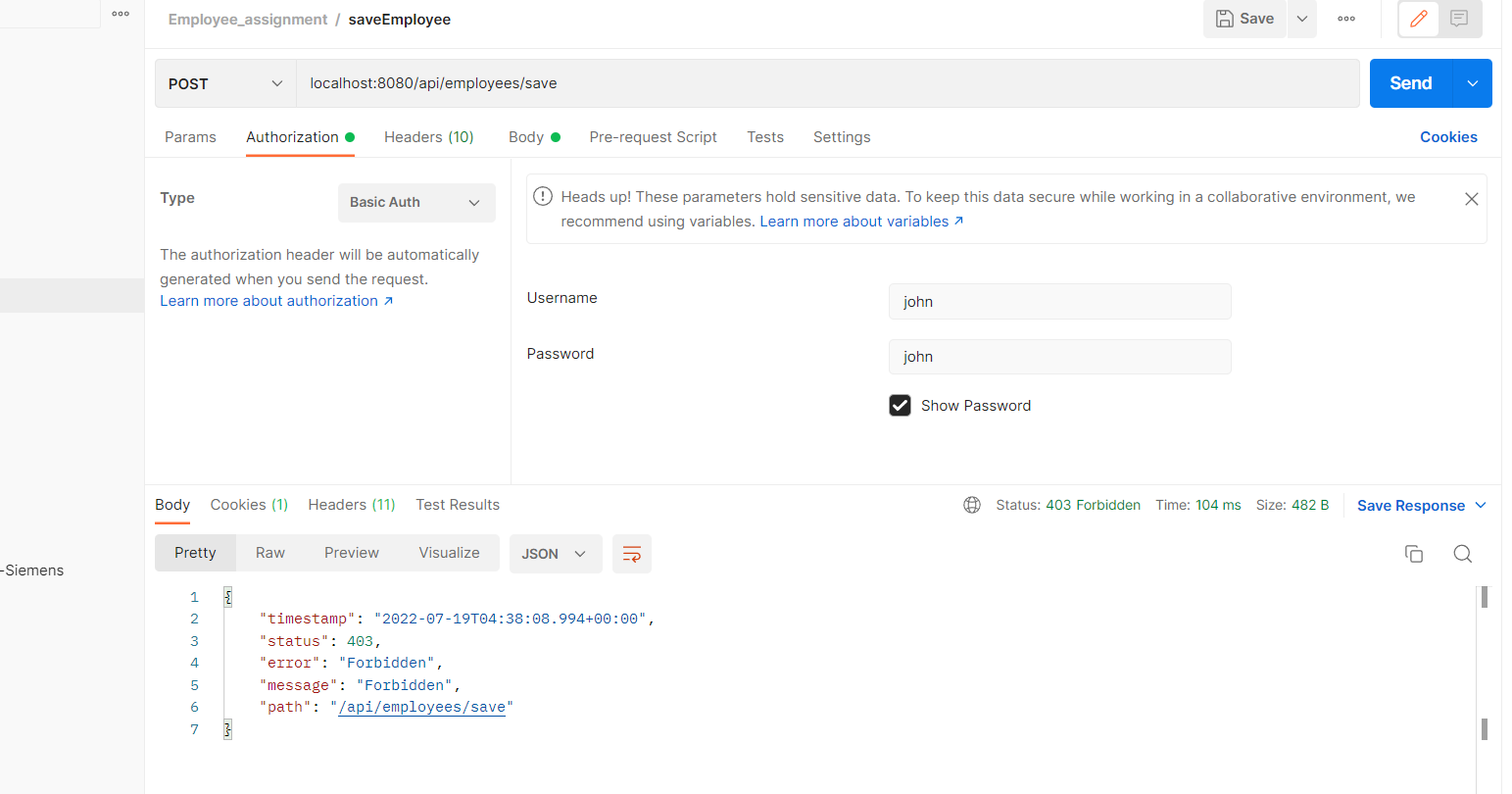
    "email": "cb@gmail.com"

}

Saving an employee with the Admin Role

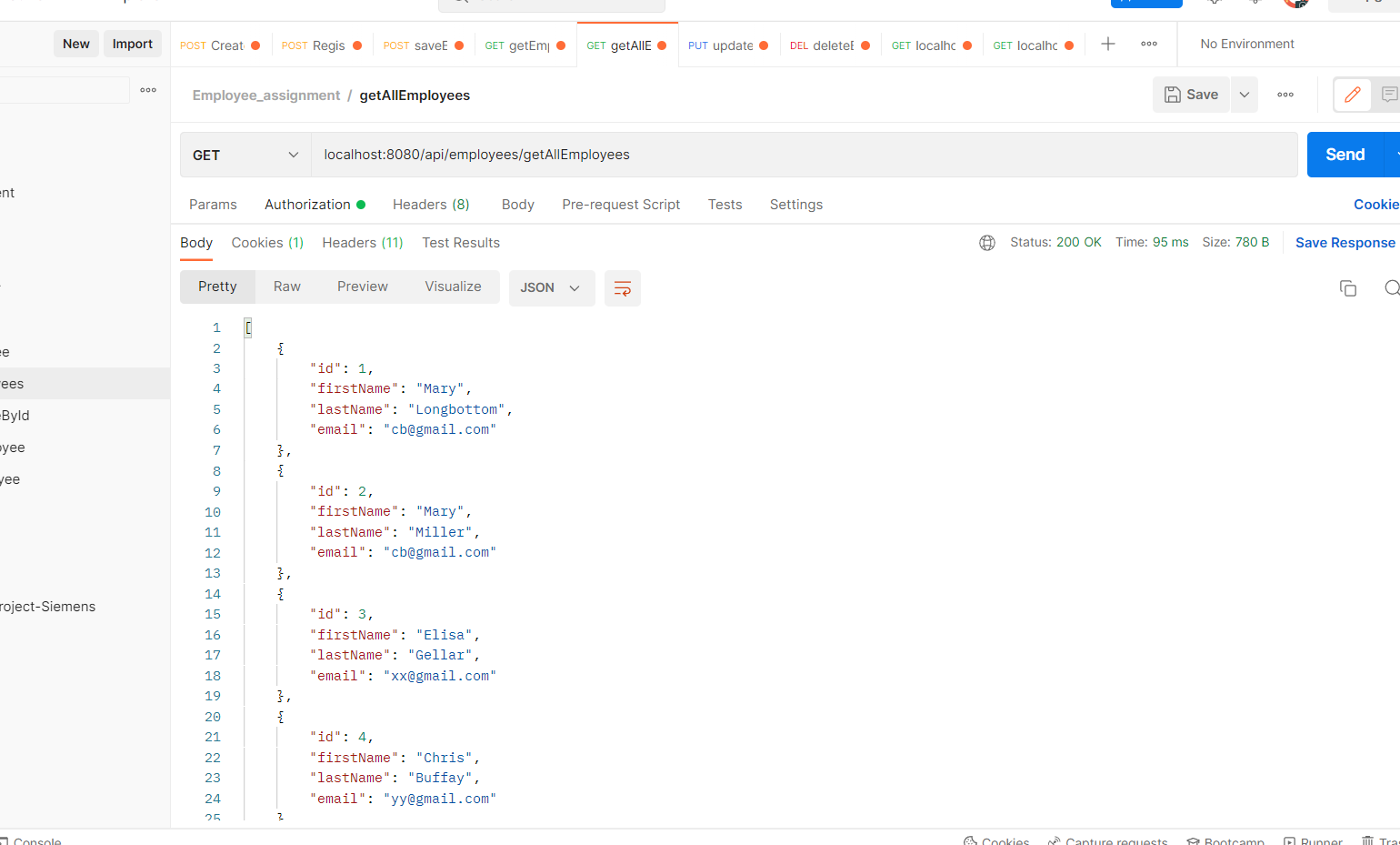


Saving an employee with the User Role gives the 403 forbidden message



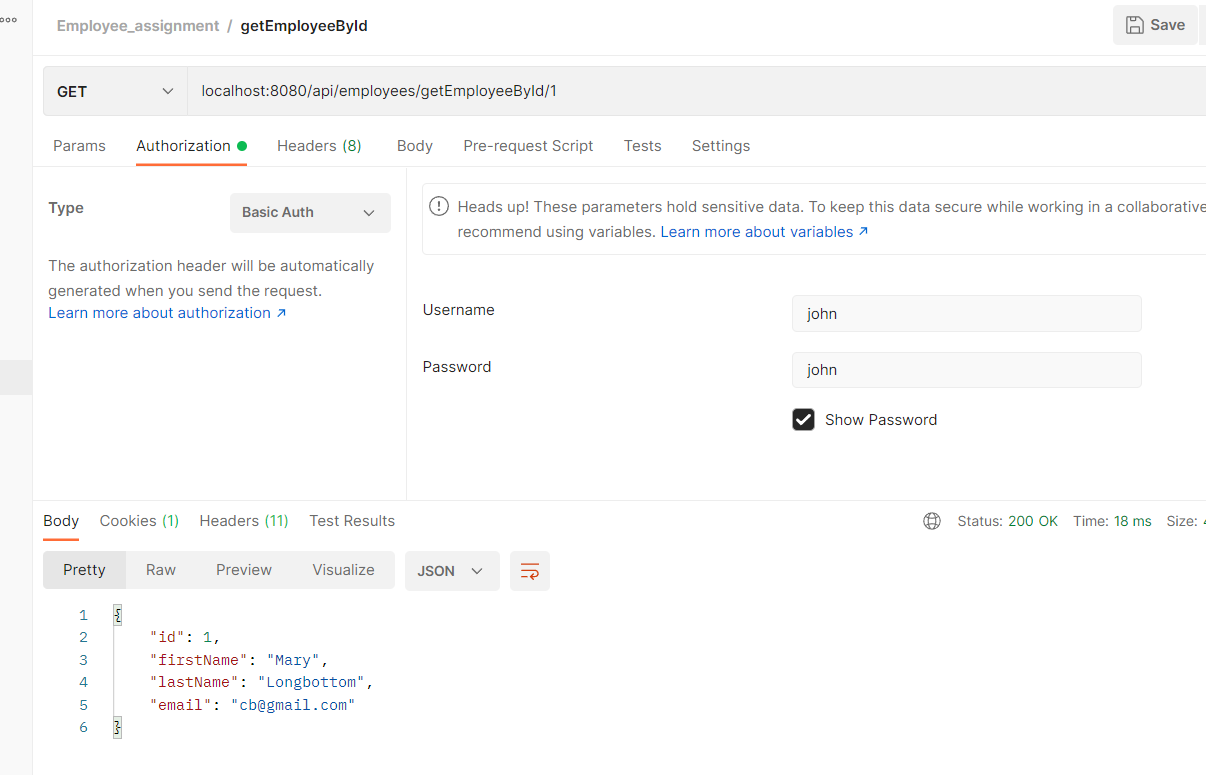
4. Your application should provide an endpoint to list all the employees stored in the database**.**

Ex-

localhost:8080/api/employees/getAllEmployees

# 5. Your application should provide endpoint to fetch or get an employee record specifically based on the id of that employee-

Ex- Url- localhost:8080/api/employees/getEmployeeById/1



# 6. Your application should provide an endpoint to update an existing employee record with the given updated json object.

Ex-

localhost:8080/api/employees/update

Request Body after updation-

 {

        "id": 1,

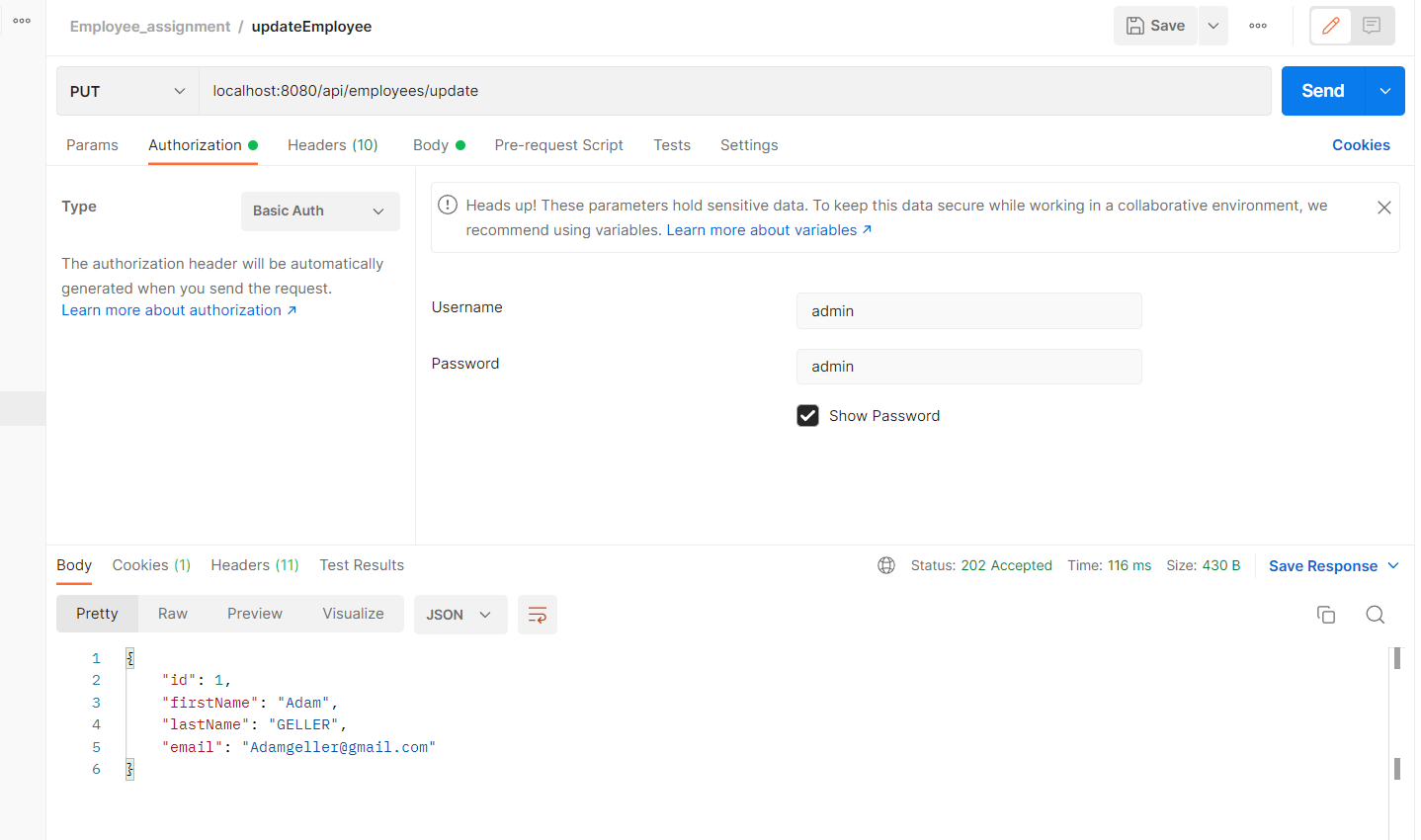
        "firstName": "Smith",

        "lastName": "Hannigan",

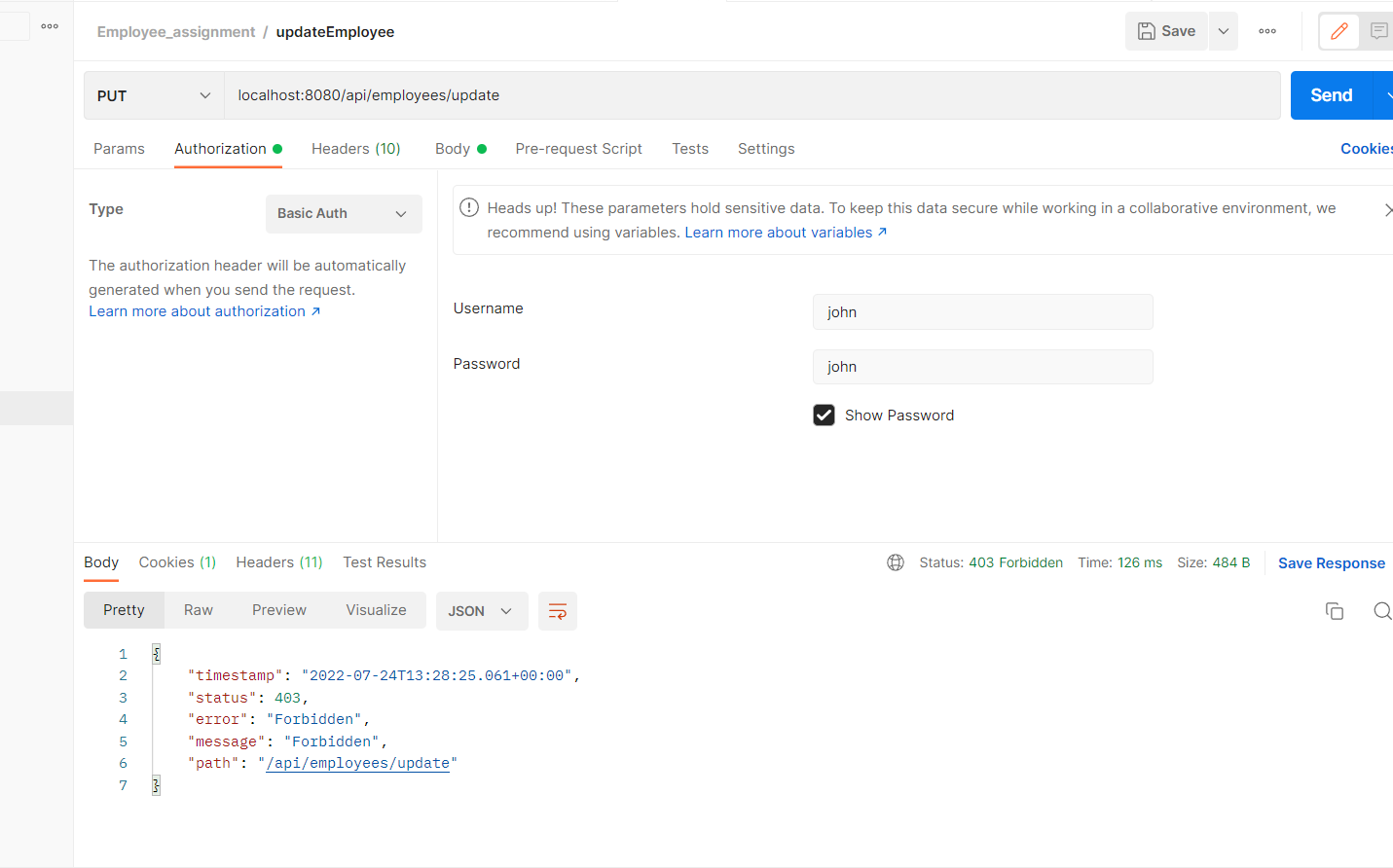
        "email": "Smith@gmail.com"

    }

Admin user can update employee’s detail successfully.



Updating employee details with user role results in 403 forbidden error.

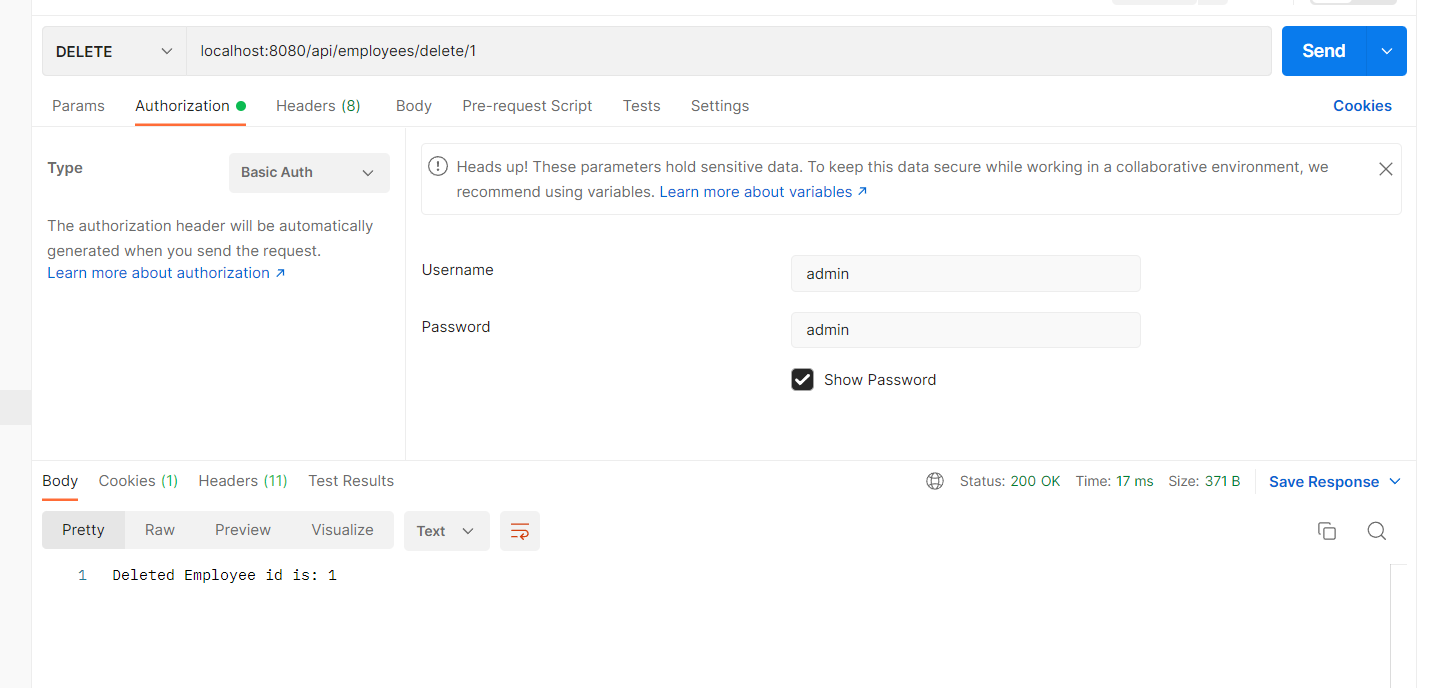


# 7. Your application should also provide an endpoint to delete an existing employee record based on the id of the employee-

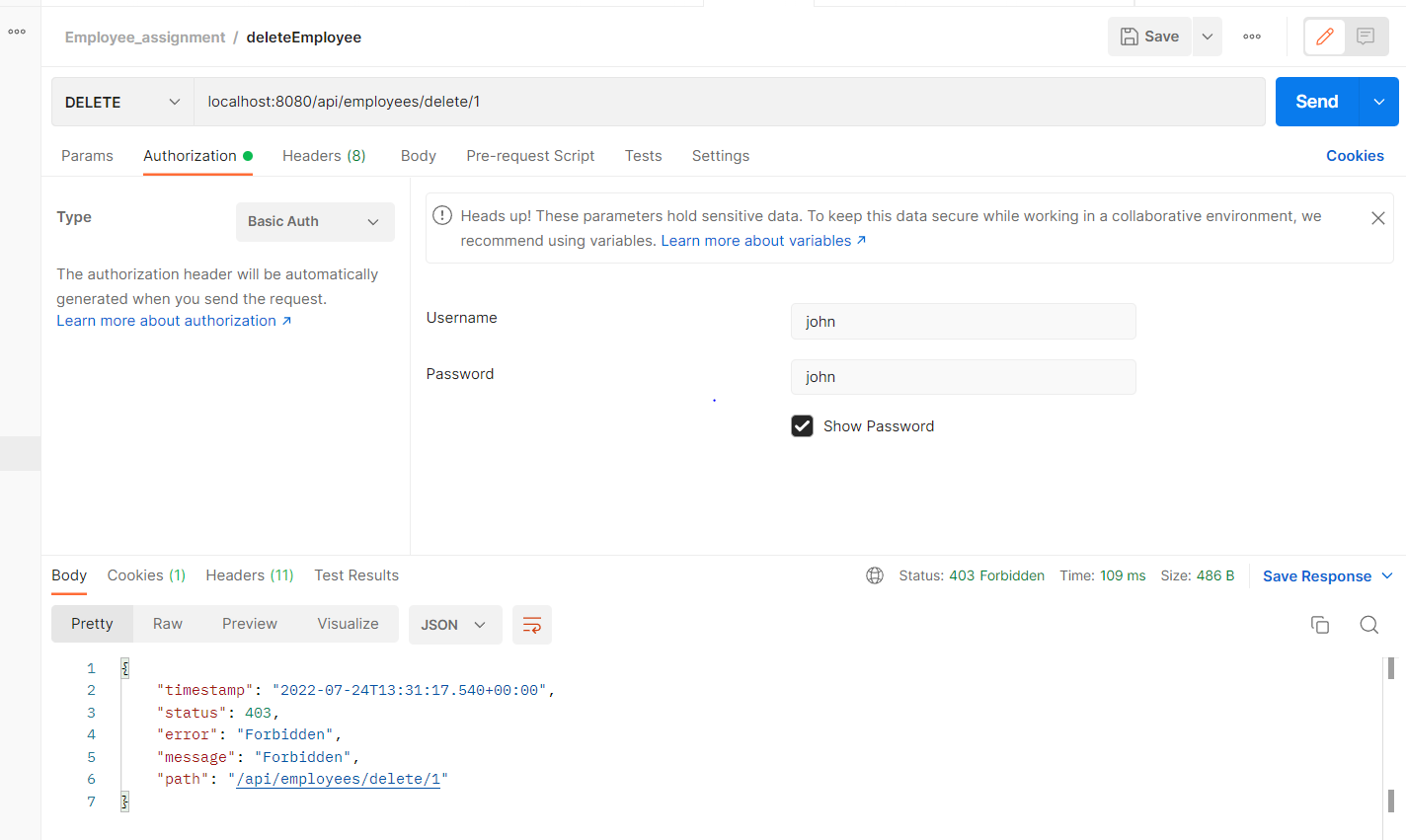
Ex-

Url- localhost:8080/api/employees/delete/1

Admin user can delete an employee’s detail successfully.



Deleting an employee with user role results in 403 forbidden error.

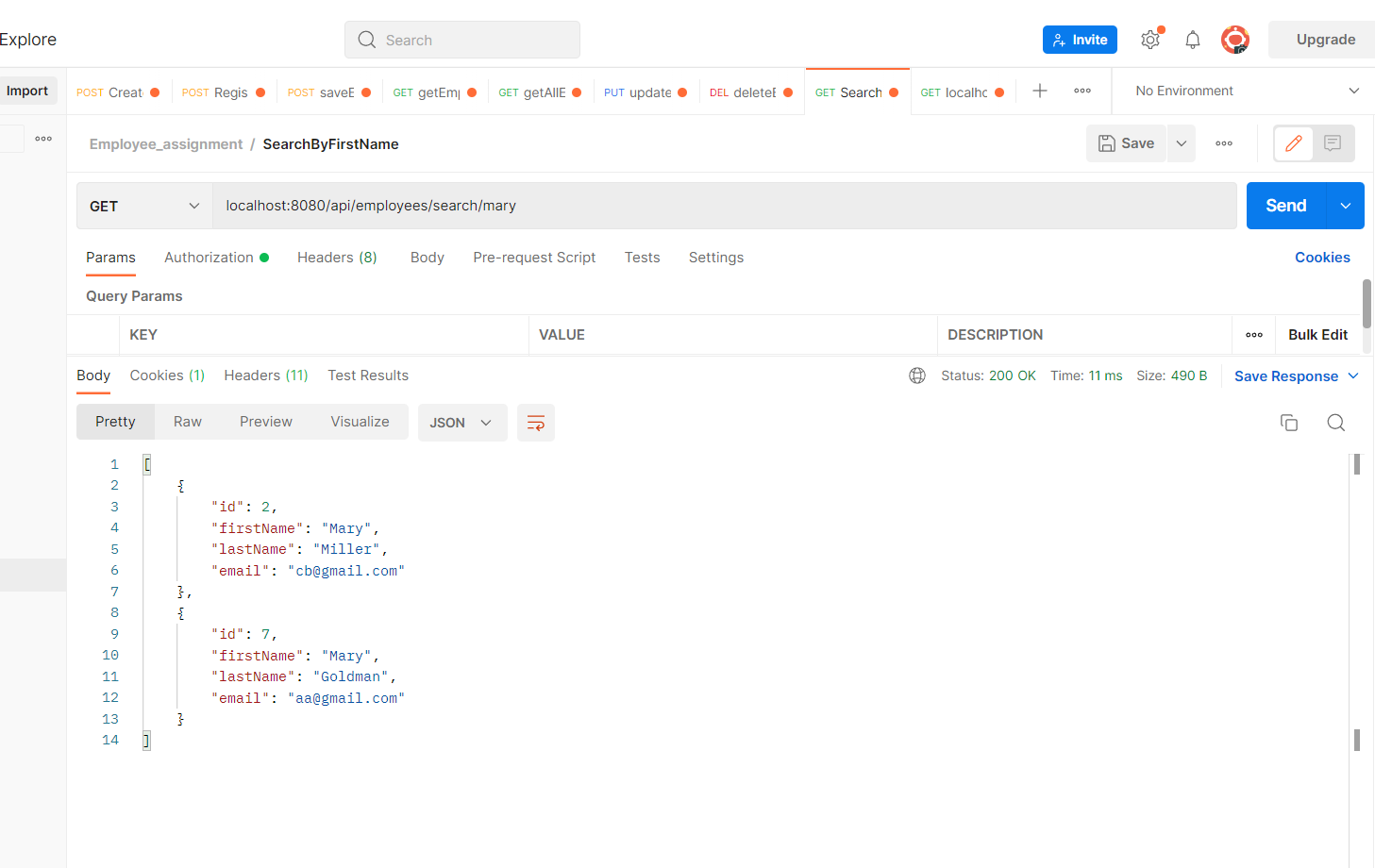


# 8. Your application should provide an endpoint to fetch an employee by his/her first name and if found more than one record then list them all-

Ex-

localhost:8080/api/employees/search/mary

Two employees with the same first name(Mary) retrieved successfully.



# 9. Your application should be able to list all employee records sorted on their first name in either ascending order or descending order .

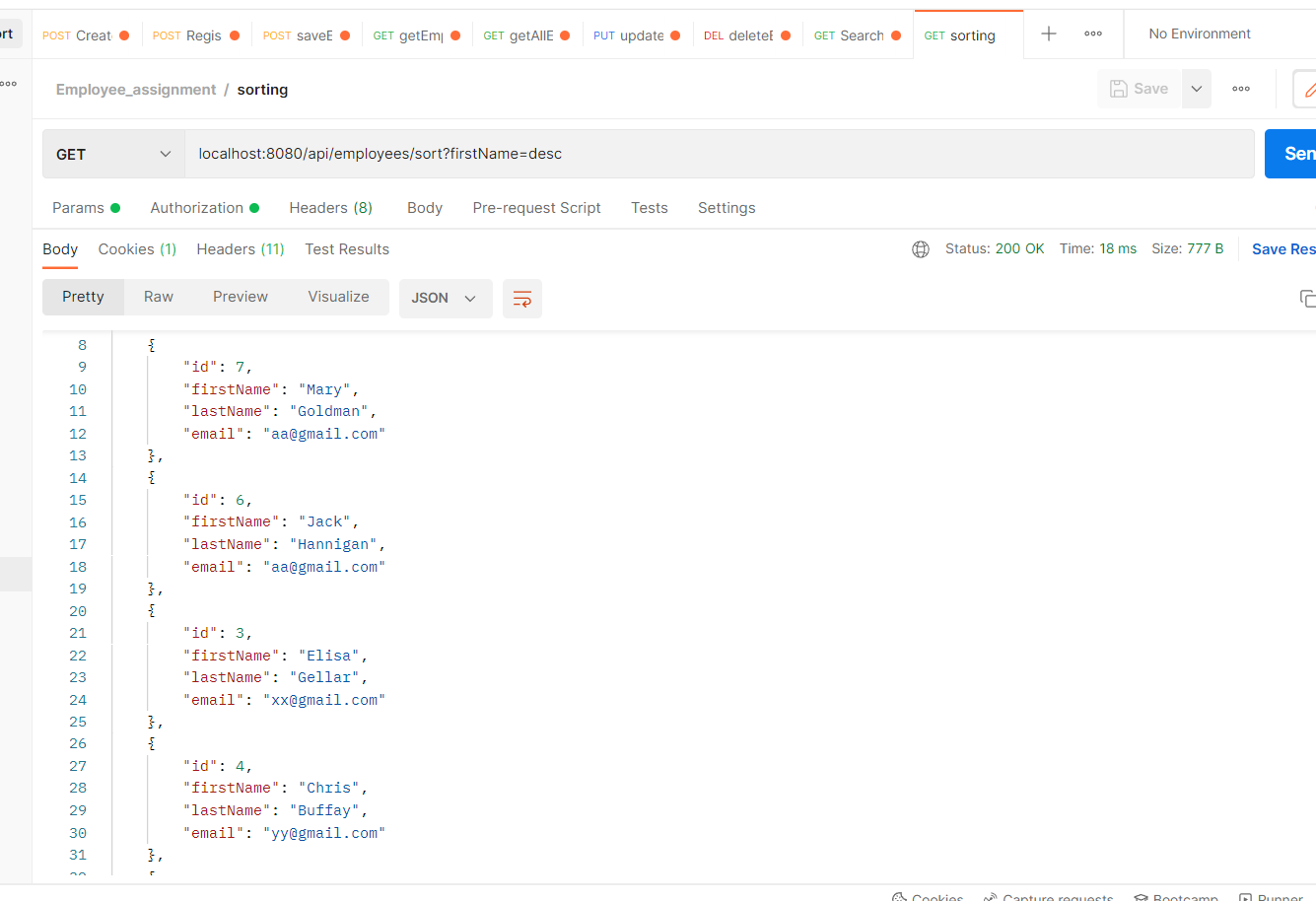
Ex-

Url- localhost:8080/api/employees/sort?firstName=desc

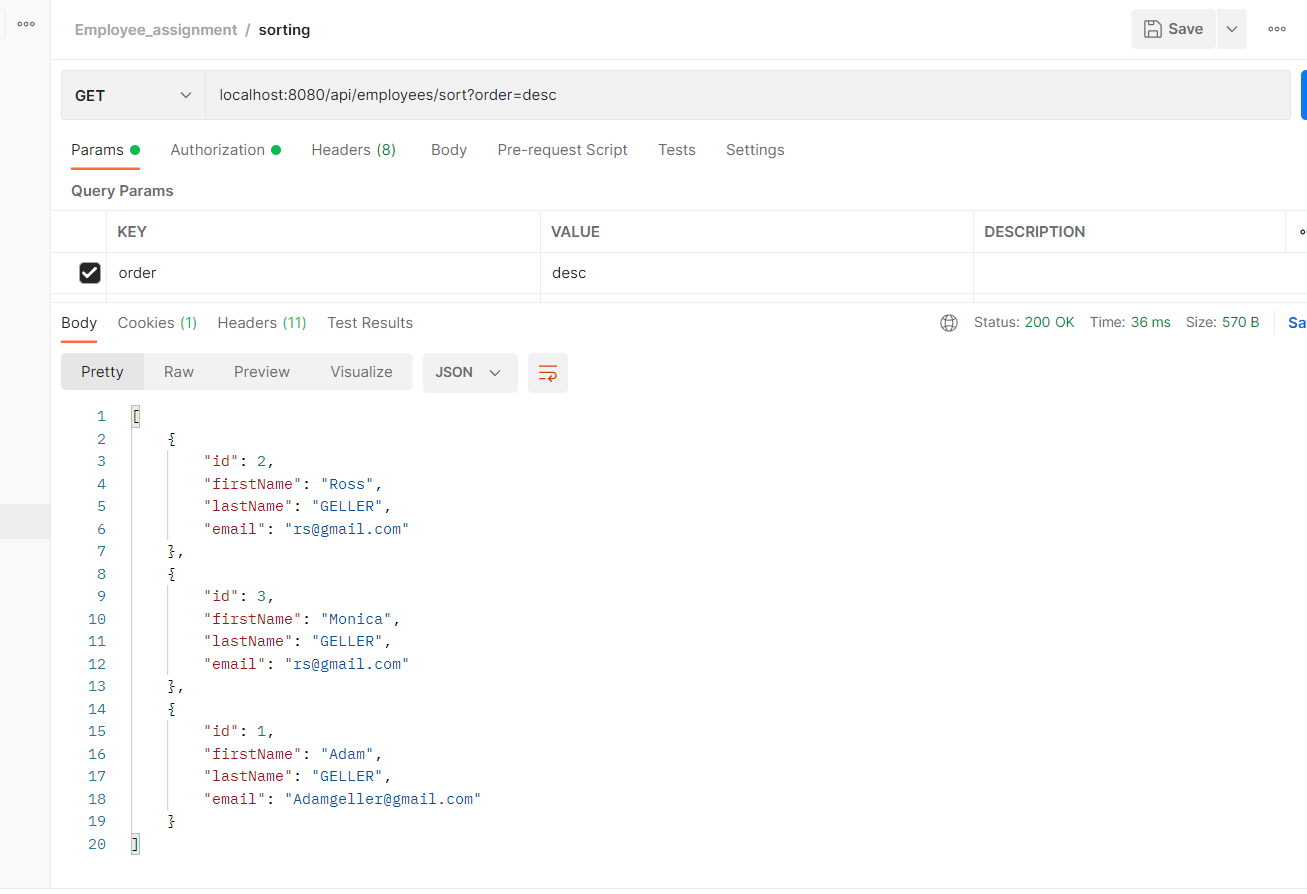
OR

Url- localhost:8080/api/employees/sort?firstName=asc

Employees are sorted based on their ‘first name’ in ascending order.



Employees are sorted based on their ‘first name’ in descending order.



-------------------------------------------------------------------------------------------------------------------------

**Important instructions**

i) You should use the H2 In Memory database for the whole project along with Spring JPA and Spring Security.

ii) Provide Screenshots of the operations(PostMan/Browser) along with code submission. (note → Screenshots will one of the criterias while grading)

iii) You can also record your screen while demonstrating CRUD operation, upload on the drive and share the drive link along with code.

iv) Spring Boot Application must follow the standard project structure .

v) Code should follow naming conventions along with proper indentations. vi) You are free to choose any Rest client to interact with api while implementation.(Prefer PostMan)