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.

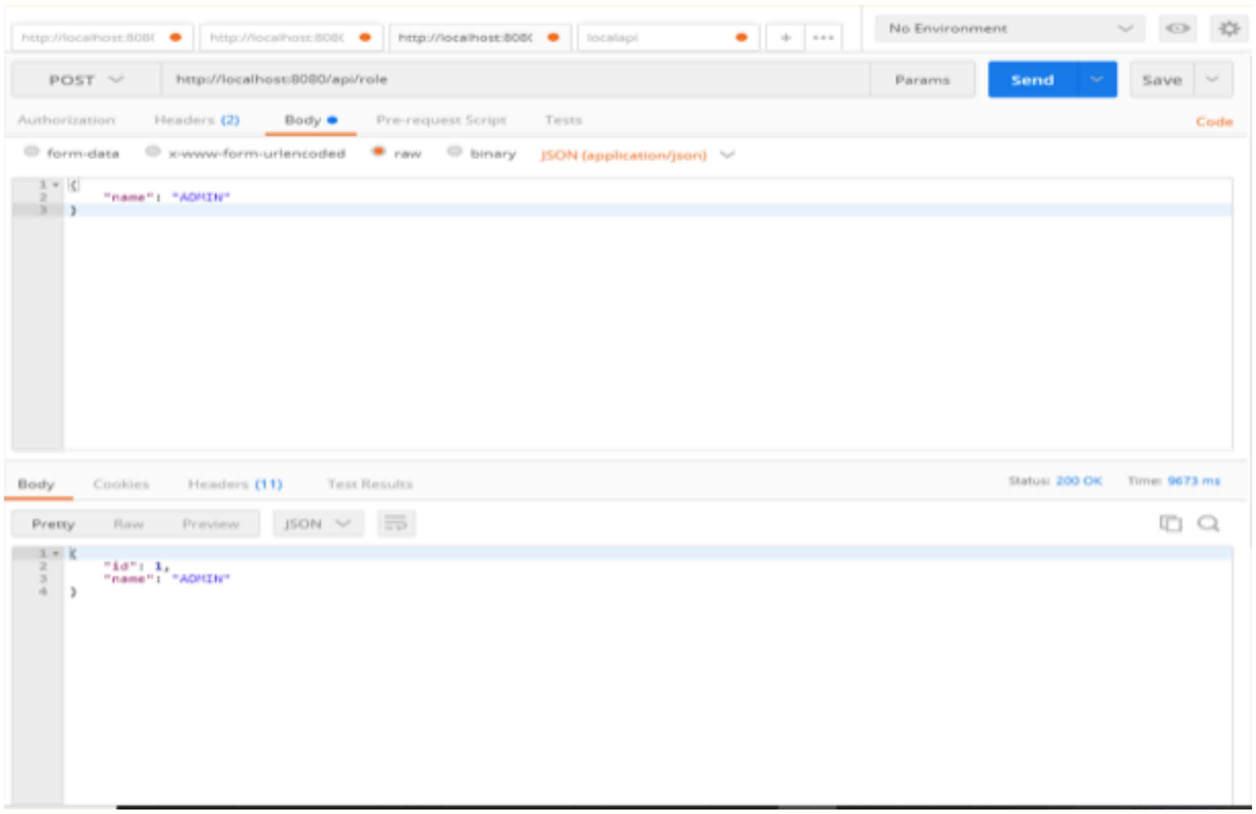
Your Rest Api should be secure.And should have different endpoints for different operations-

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

Ex-

```
{
    "name":"USER"
}
```

Where name specifies a role which can be assigned to a user that will be used for authentication purposes while interacting with the api.



POST http://localhost:8080/api/role

Body: raw, JSON (application/json)

```
{
    "name": "ADMIN"
}
```

Status: 200 OK Time: 9673 ms

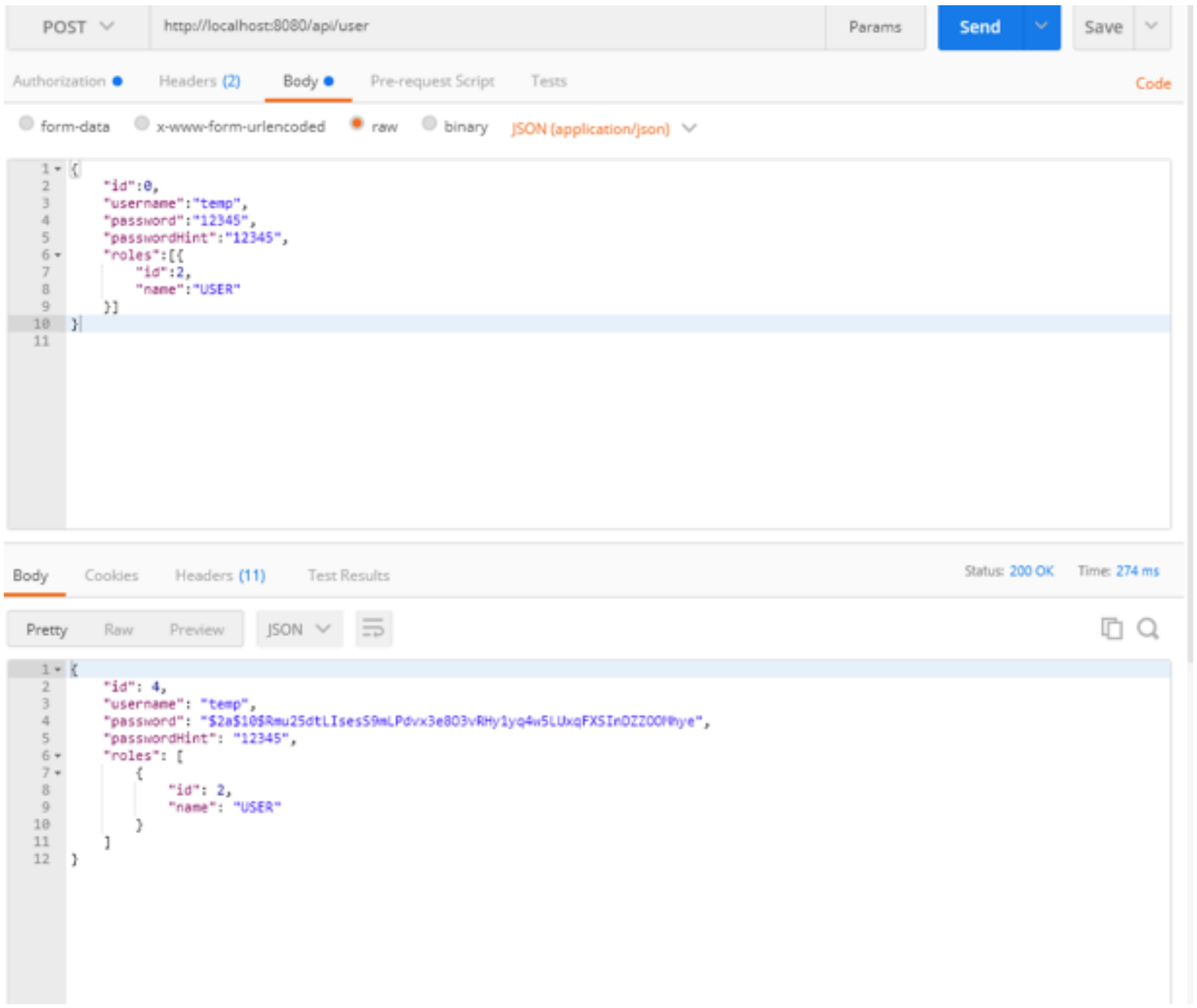
Body: Pretty, JSON

```
{
    "id": 1,
    "name": "ADMIN"
}
```

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

Ex-

```
{
    "username":"temp",
    "password":"12345",
    "roles":[{
        "id":2,
        "name":"USER"
    }]
}
```



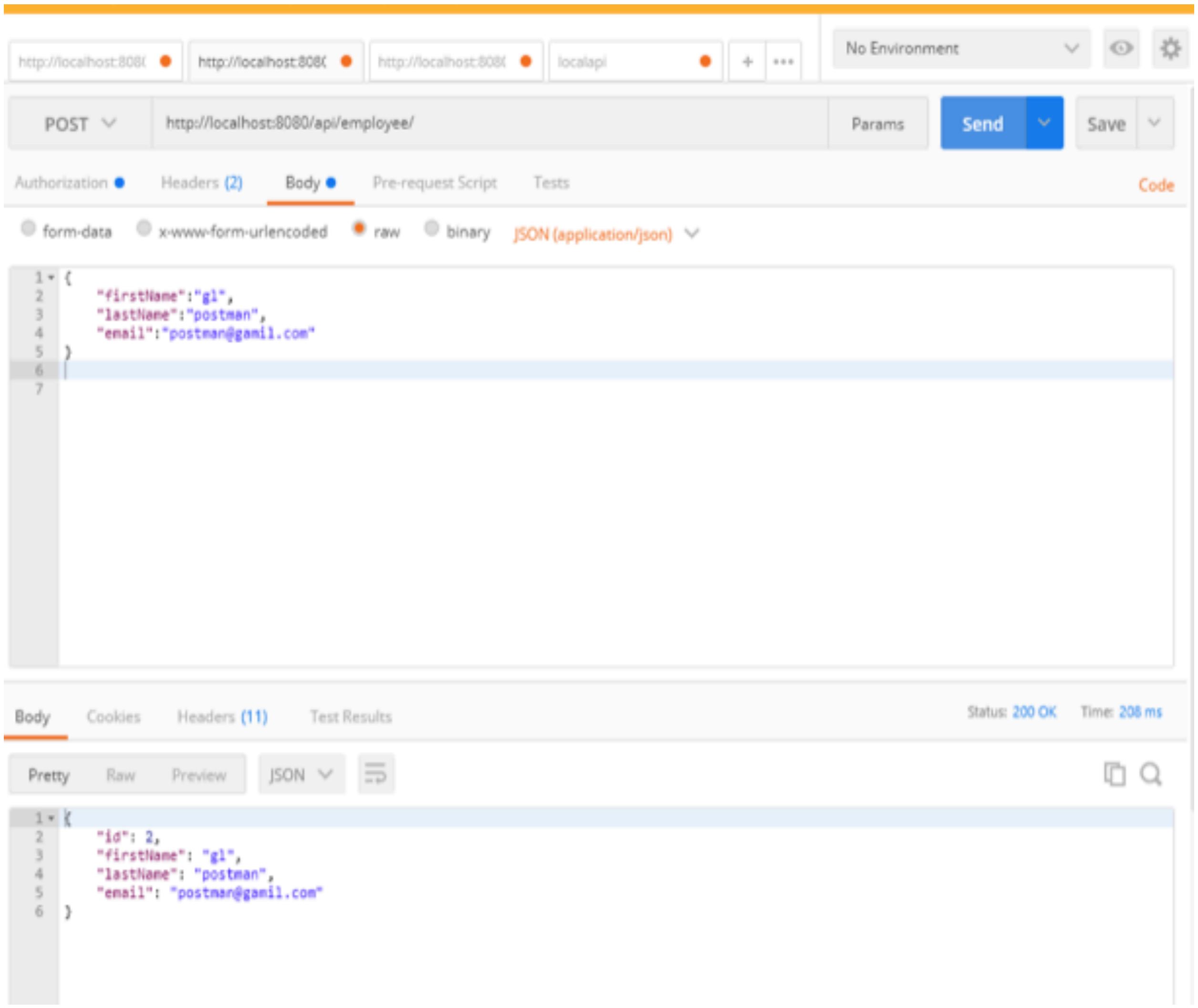
```
{
    "id":0,
    "username":"temp",
    "password":"12345",
    "passwordHint":"12345",
    "roles":[{
        "id":2,
        "name":"USER"
    }]
}
```

```
{
    "id": 4,
    "username": "temp",
    "password": "$2a$10$Rmu25dtLIsesS9mLPdvx3e8O3vRHy1yq4w5LUxqFXSInDZZOOMye",
    "passwordHint": "12345",
    "roles": [
        {
            "id": 2,
            "name": "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":"gl",
    "lastName":"postman",
    "email":"postman@gamil.com"
}
```



http://localhost:8080/api/employee/

JSON (application/json)

```
{
    "firstName":"gl",
    "lastName":"postman",
    "email":"postman@gamil.com"
}
```

Status: 200 OK   Time: 208 ms

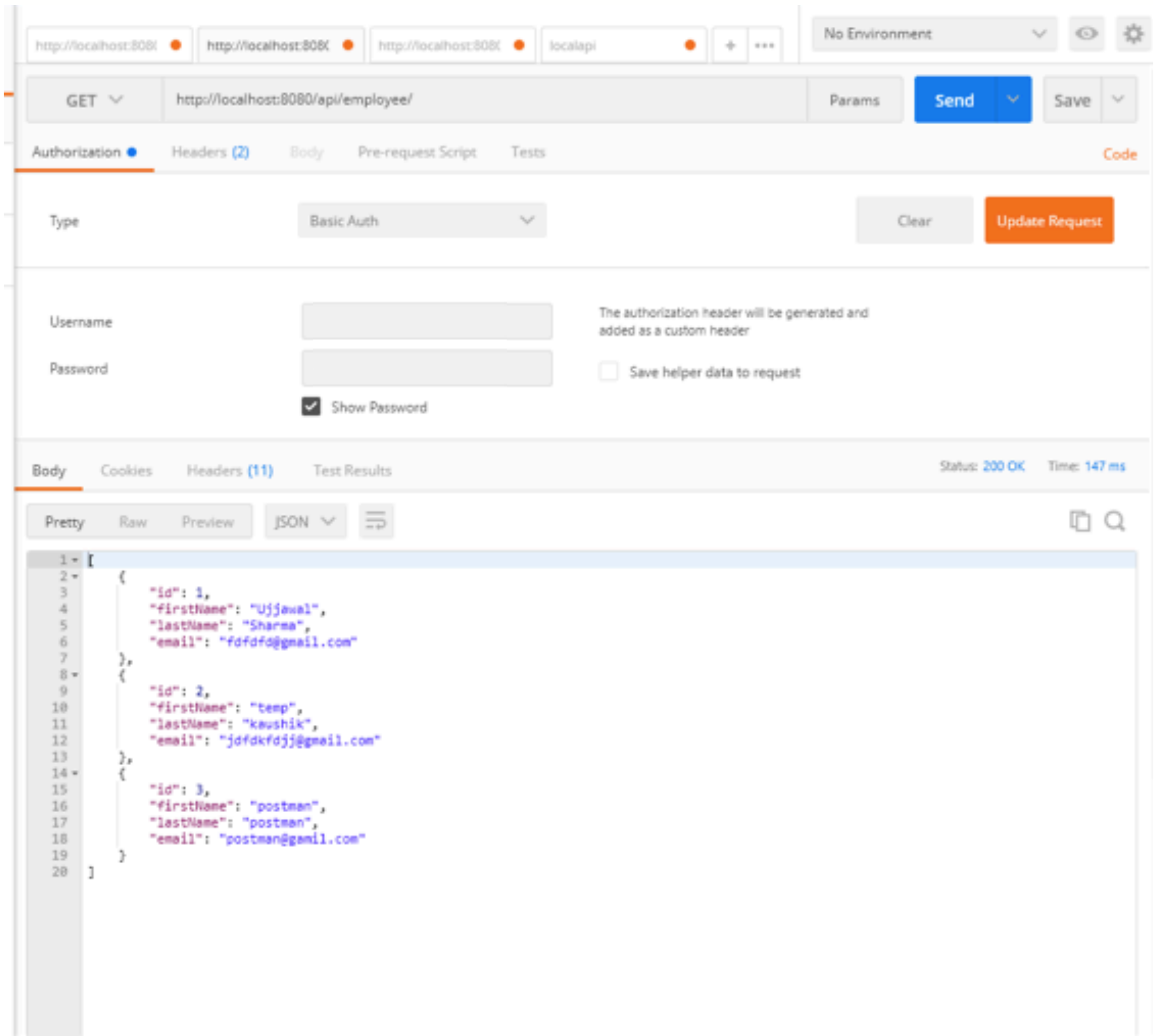
```
{
    "id": 2,
    "firstName": "gl",
    "lastName": "postman",
    "email": "postman@gamil.com"
}
```

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

Ex-

Response Body-

```
[
    {
        "id": 1,
        "firstName": "Ujjawal",
        "lastName": "Sharma",
        "email": "fdfdfd@gmail.com"
```



GET http://localhost:8080/api/employee/

Type: Basic Auth

Username

Password

Status: 200 OK Time: 147 ms

```
[
    {
        "id": 1,
        "firstName": "Ujjawal",
        "lastName": "Sharma",
        "email": "fdfdfd@gmail.com"
    },
    {
        "id": 2,
        "firstName": "temp",
        "lastName": "kaushik",
        "email": "jdfdkfdjj@gmail.com"
    },
    {
        "id": 3,
        "firstName": "postman",
        "lastName": "postman",
        "email": "postman@gamil.com"
    }
]
```

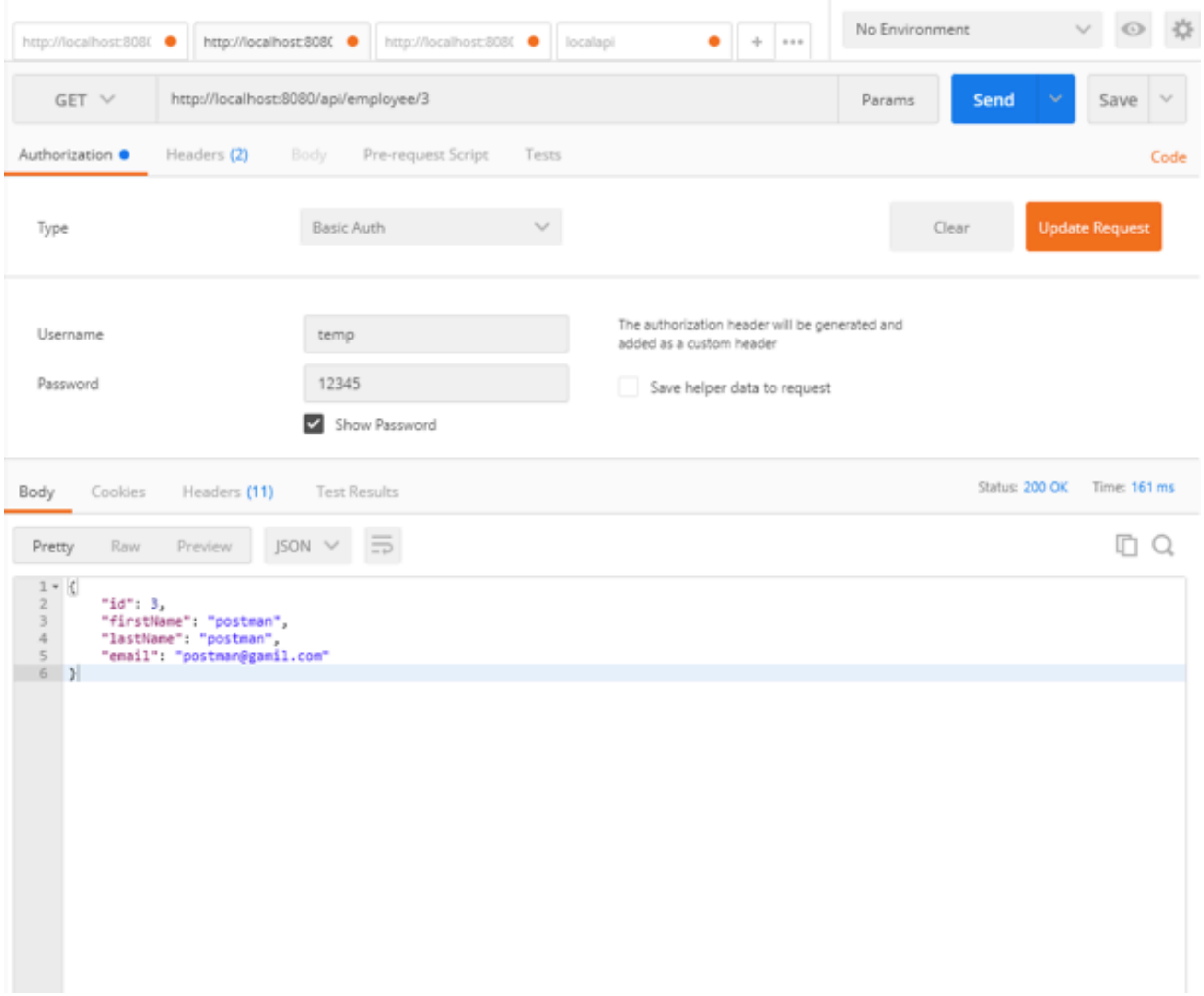
```
    },
    {
        "id": 2,
        "firstName": "temp",
        "lastName": "kaushik",
        "email": "jdfdkfdjj@gmail.com"
    },
    {
        "id": 3,
        "firstName": "postman",
        "lastName": "postman",
        "email": "postman@gamil.com"
    }
]
```

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

Ex- Url- http://localhost:8080/api/employees/3

Response Body-

```
{
    "id": 3,
    "firstName": "postman",
    "lastName": "postman",
    "email": "postman@gamil.com"
}
```



GET http://localhost:8080/api/employee/3

Type: Basic Auth

Username: temp

Password: 12345

[✓] Show Password

The authorization header will be generated and added as a custom header

Save helper data to request

Status: 200 OK Time: 161 ms

```
{
    "id": 3,
    "firstName": "postman",
    "lastName": "postman",
    "email": "postman@gamil.com"
}
```

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

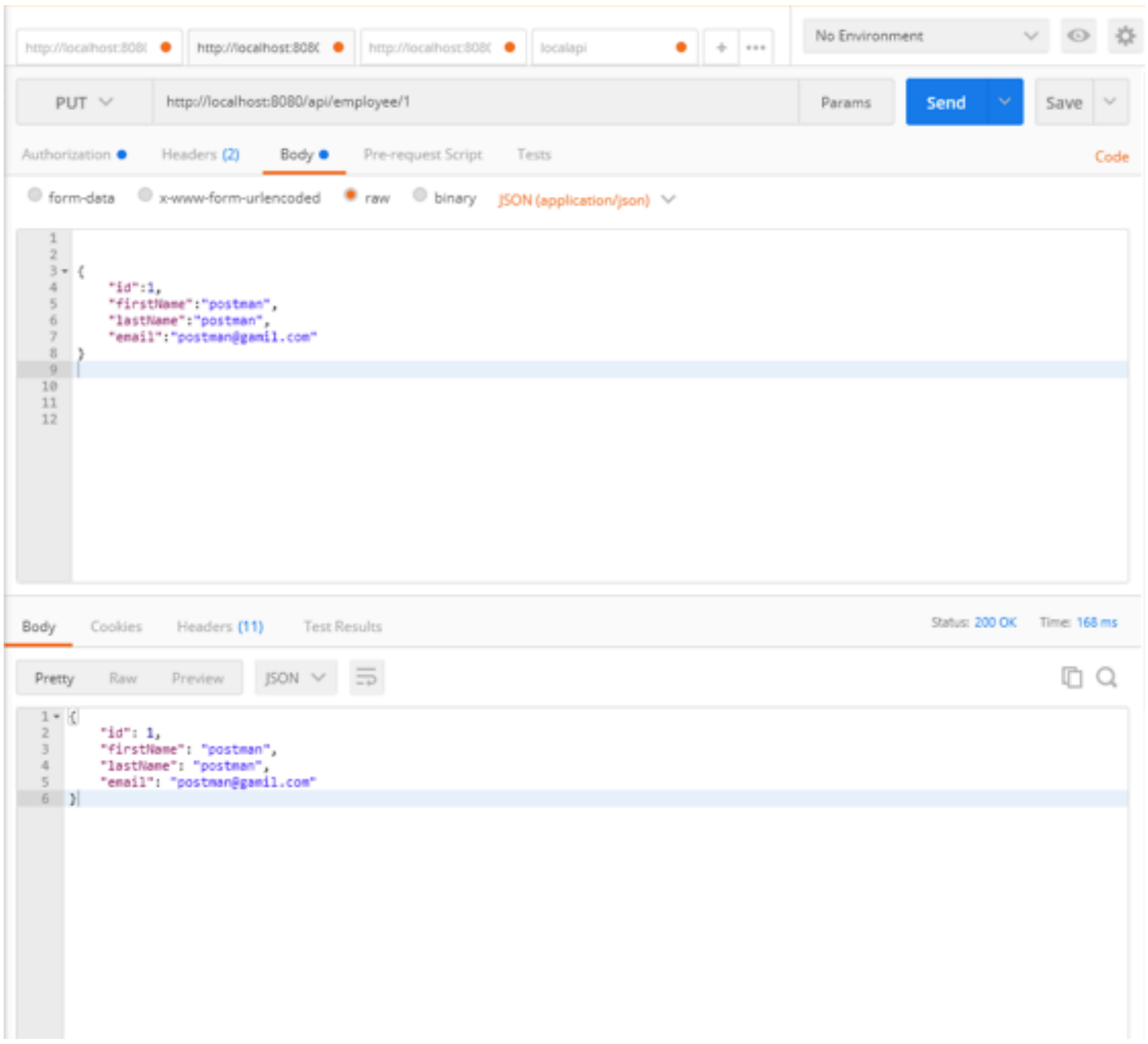
Ex-

Object to be updated(raw->Json)-

```
{
    "id":1,
    "firstName":"postman",
    "lastName":"postman",
    "email":"postman@gamil.com"
}
```

Response Body after updation-

```
{
    "id": 1,
    "firstName": "postman",
    "lastName": "postman",
    "email": "postman@gamil.com"
}
```



PUT http://localhost:8080/api/employee/1

Body: raw, JSON (application/json)

```
{
    "id":1,
    "firstName":"postman",
    "lastName":"postman",
    "email":"postman@gamil.com"
}
```

Status: 200 OK Time: 168 ms

```
{
    "id": 1,
    "firstName": "postman",
    "lastName": "postman",
    "email": "postman@gamil.com"
}
```

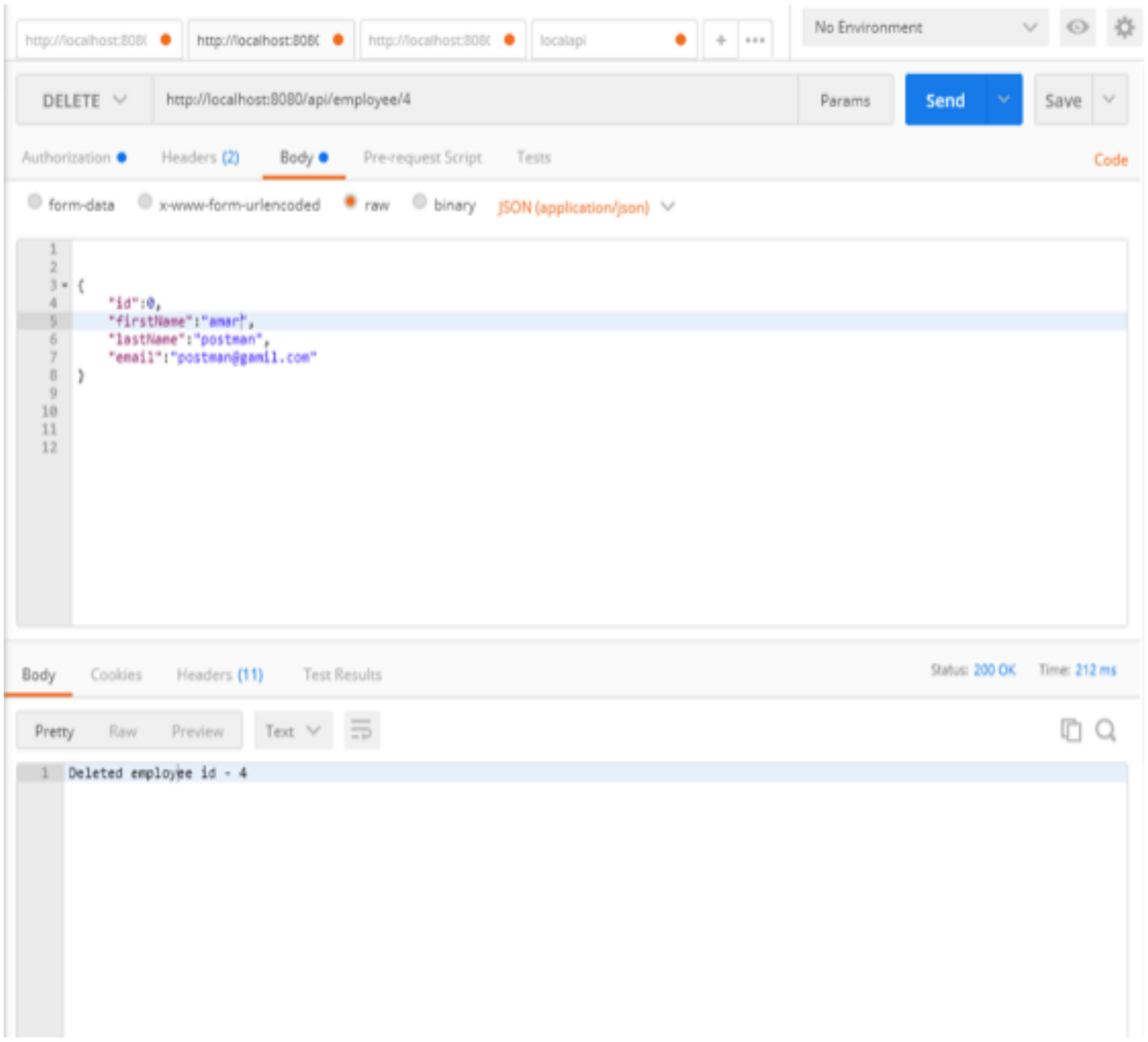
7. Your application should also provide an endpoint to delete an existing employee record based on the id of the employee-

Ex-

Url- http://localhost:8080/api/employees/4

Response Body-

```
"Deleted employee id - 4"
```



DELETE http://localhost:8080/api/employee/4

```


{
    "id":0,
    "firstName":"amar",
    "lastName":"postman",
    "email":"postman@gamil.com"
}
```

Status: 200 OK Time: 212 ms

```
Deleted employee id - 4
```

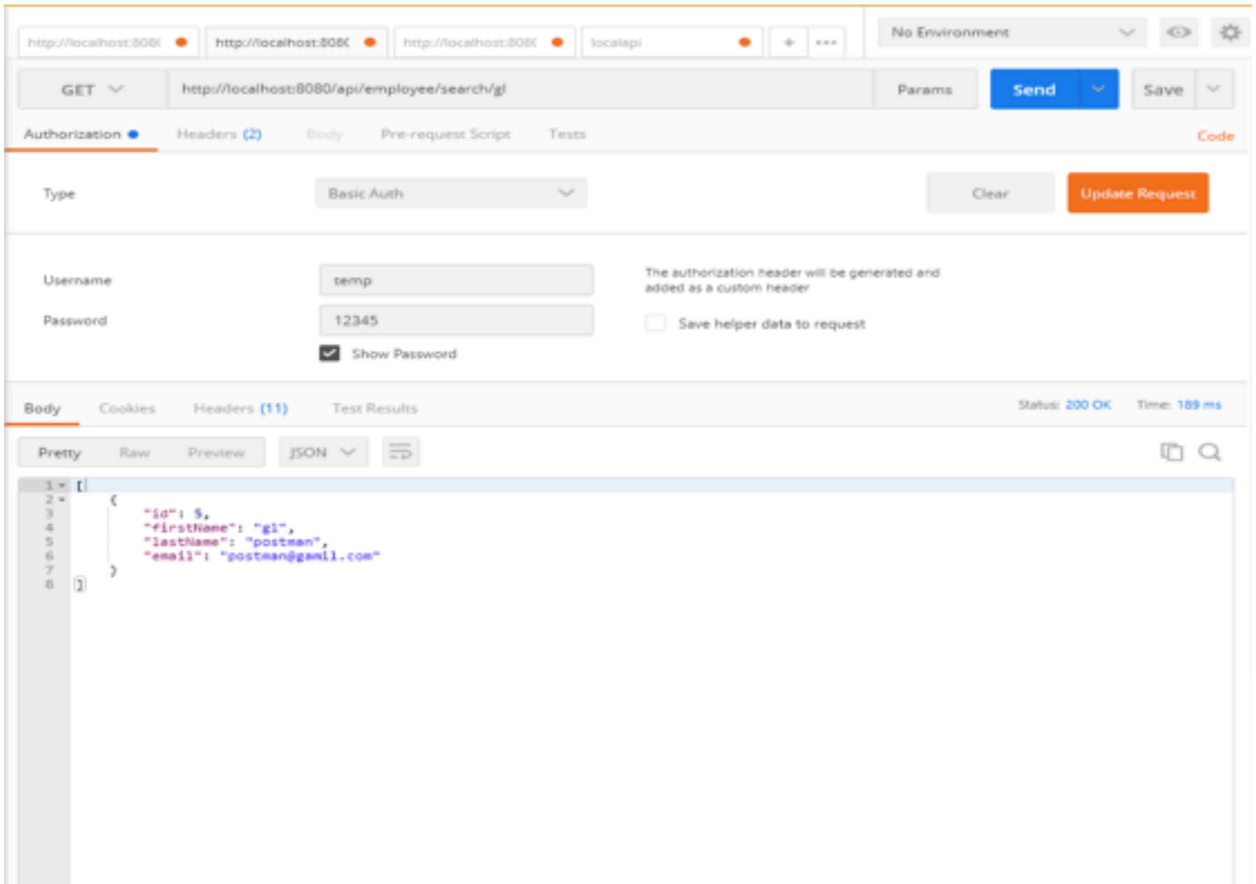
8. Your application should provide an endpoint to fetch an employee by his/ her first name and if found more than one record then list them all-

Ex-

Url- http://localhost:8080/api/employees/search/gl

Response Body-

```
[
    {
        "id": 11,
        "firstName": "gl",
        "lastName": "postman",
        "email": "postman@gamil.com"
    }
]
```



GET http://localhost:8080/api/employee/search/gl

Type: Basic Auth

Username: temp

Password: 12345

✓ Show Password

The authorization header will be generated and added as a custom header

Save helper data to request

Status: 200 OK   Time: 189 ms

```
[
    {
        "id": 5,
        "firstName": "gl",
        "lastName": "postman",
        "email": "postman@gamil.com"
    }
]
```

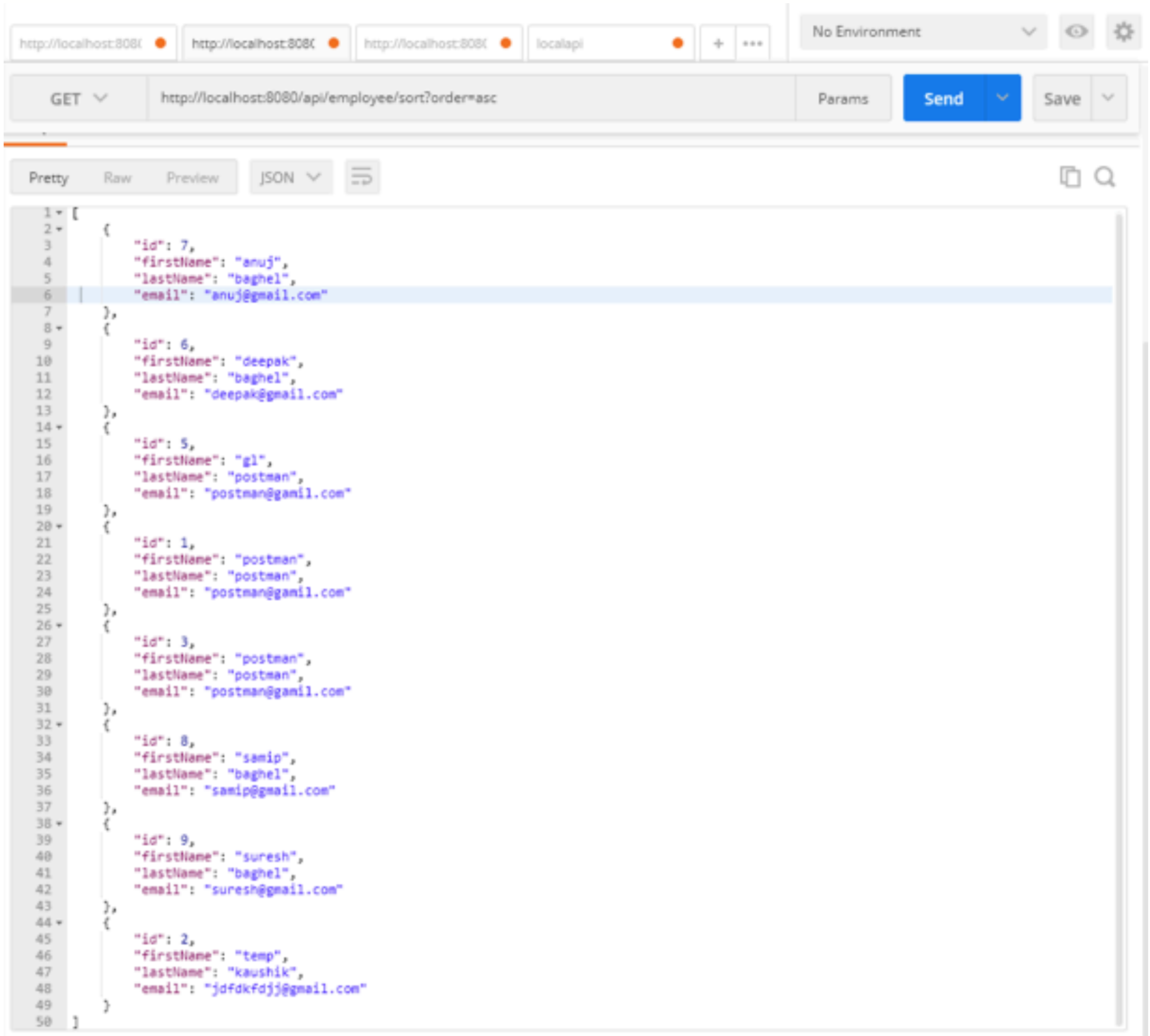
9. Your application should be able to list all employee records sorted on their first name in either ascending order or descending order .

Ex-

Url- [http://localhost:8080/api/employees/sort?order="asc](http://localhost:8080/api/employees/sort?order=”asc)”

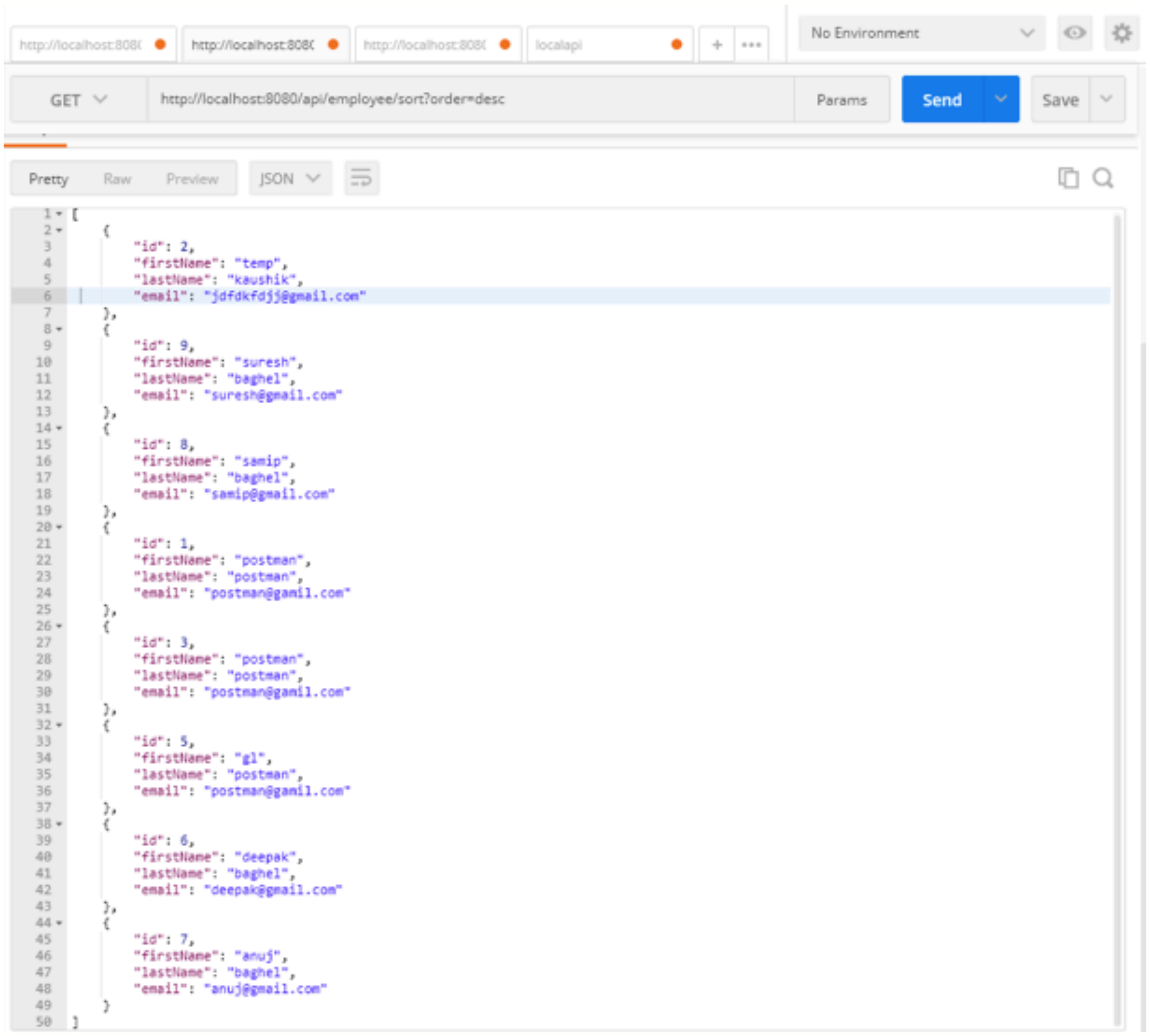
OR

Url- http://localhost:8080/api/employees/sort?order="desc”



```
http://localhost:8080/api/employee/sort?order=asc
```

```
[
    {
        "id": 7,
        "firstName": "anuj",
        "lastName": "baghel",
        "email": "anuj@gmail.com"
    },
    {
        "id": 6,
        "firstName": "deepak",
        "lastName": "baghel",
        "email": "deepak@gmail.com"
    },
    {
        "id": 5,
        "firstName": "gl",
        "lastName": "postman",
        "email": "postman@gamil.com"
    },
    {
        "id": 1,
        "firstName": "postman",
        "lastName": "postman",
        "email": "postman@gamil.com"
    },
    {
        "id": 3,
        "firstName": "postman",
        "lastName": "postman",
        "email": "postman@gamil.com"
    },
    {
        "id": 8,
        "firstName": "samip",
        "lastName": "baghel",
        "email": "samip@gmail.com"
    },
    {
        "id": 9,
        "firstName": "suresh",
        "lastName": "baghel",
        "email": "suresh@gmail.com"
    },
    {
        "id": 2,
        "firstName": "temp",
        "lastName": "kaushik",
        "email": "jdfdkfdjj@gmail.com"
    }
]
```



```
GET  http://localhost:8080/api/employee/sort?order=desc
```

```
[
    {
        "id": 2,
        "firstName": "temp",
        "lastName": "kaushik",
        "email": "jdfdkfdjj@gmail.com"
    },
    {
        "id": 9,
        "firstName": "suresh",
        "lastName": "baghel",
        "email": "suresh@gmail.com"
    },
    {
        "id": 8,
        "firstName": "samip",
        "lastName": "baghel",
        "email": "samip@gmail.com"
    },
    {
        "id": 1,
        "firstName": "postman",
        "lastName": "postman",
        "email": "postman@gamil.com"
    },
    {
        "id": 3,
        "firstName": "postman",
        "lastName": "postman",
        "email": "postman@gamil.com"
    },
    {
        "id": 5,
        "firstName": "gl",
        "lastName": "postman",
        "email": "postman@gamil.com"
    },
    {
        "id": 6,
        "firstName": "deepak",
        "lastName": "baghel",
        "email": "deepak@gmail.com"
    },
    {
        "id": 7,
        "firstName": "anuj",
        "lastName": "baghel",
        "email": "anuj@gmail.com"
    }
]
```

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)