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.

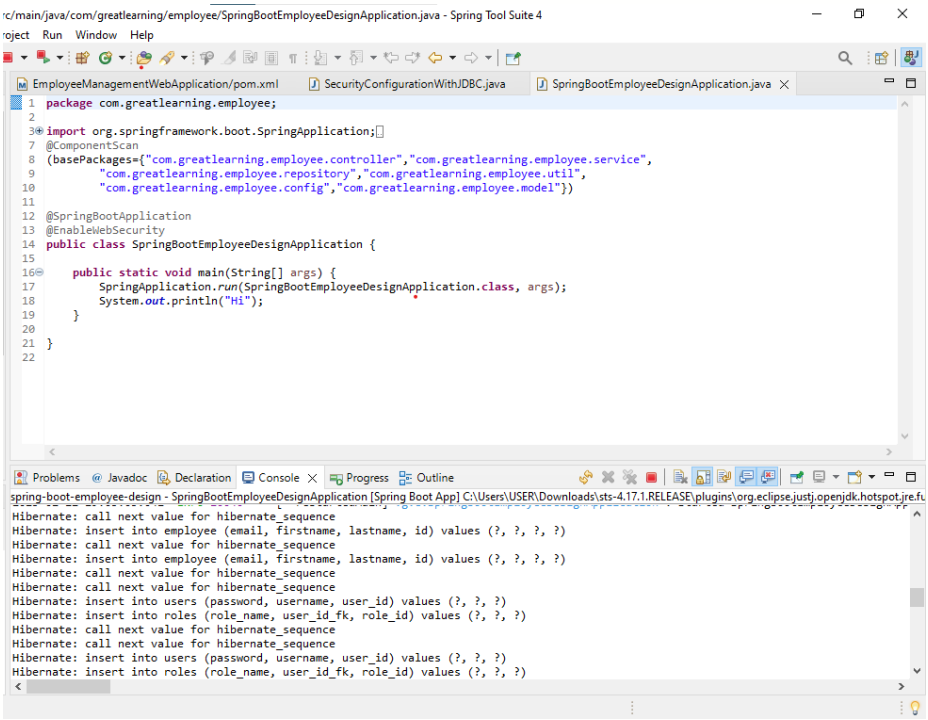
Your Rest API should be secure. And should have different endpoints for different operations-

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

Ex-

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

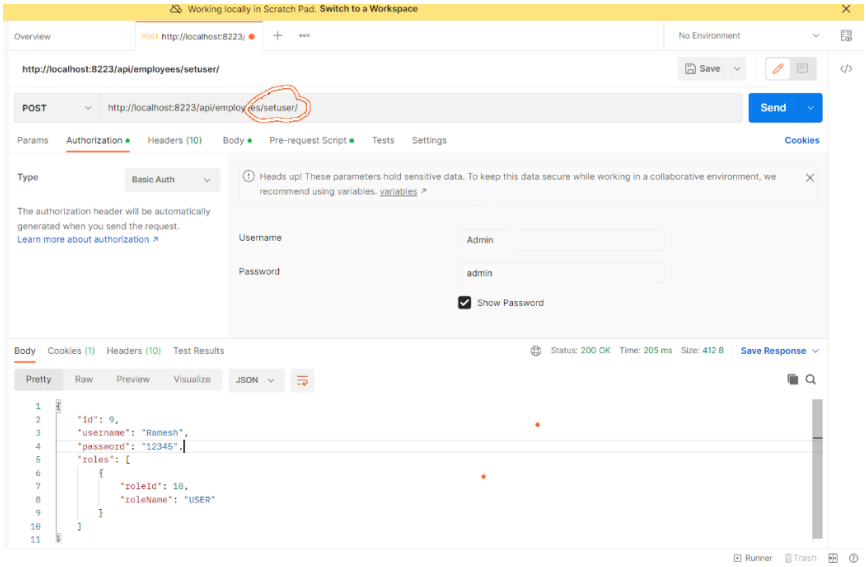
Where name specifies a role which can be assigned to a user that will be used for authentication purposes while interacting with the API.



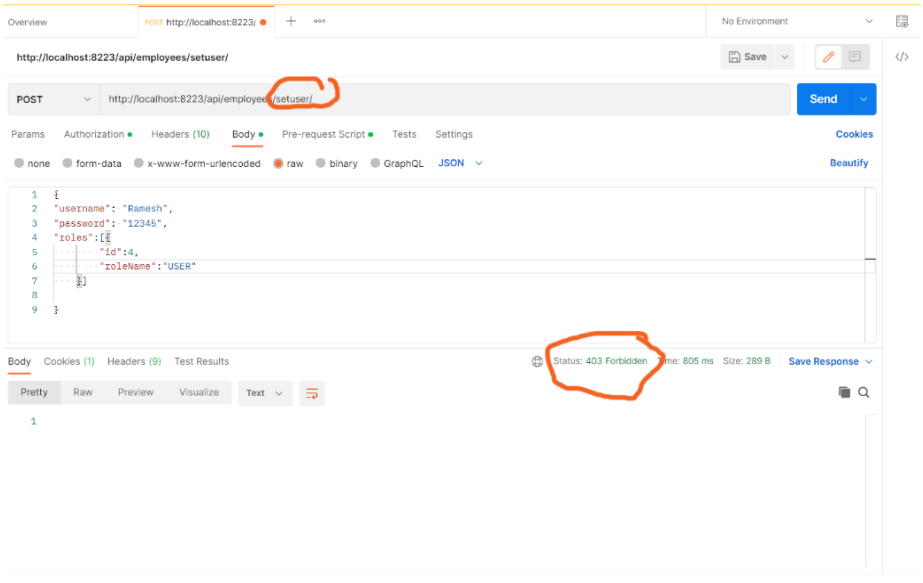
2. Your application should be able to add Users in the db which can be used for authentication purposes.

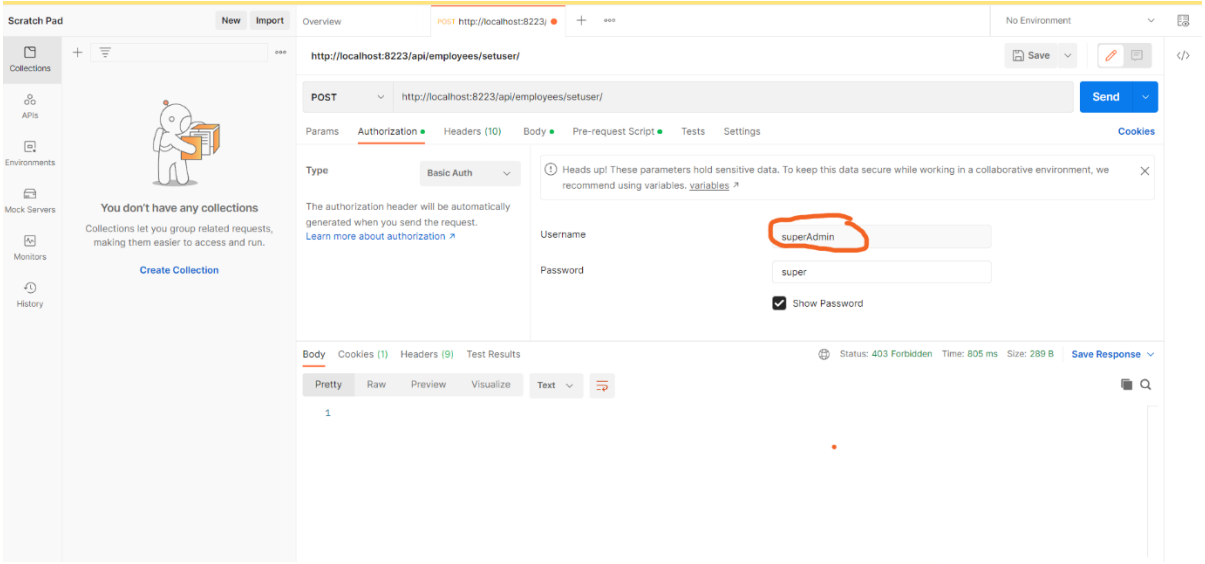
Ex-

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

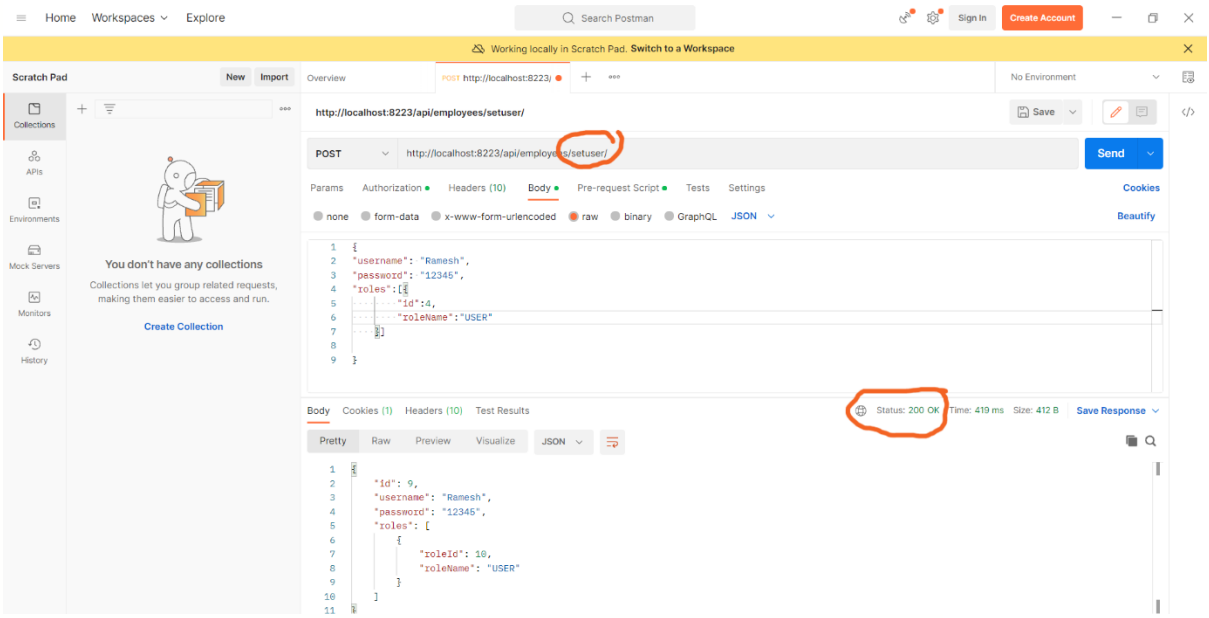



USERNAME AS ADMIN DOES NOT HAVE THE AUTHORISATION TO SET USERS.





USER AS SUPERADMIN HAS THE AUTHORISATION TO ADD USERS.

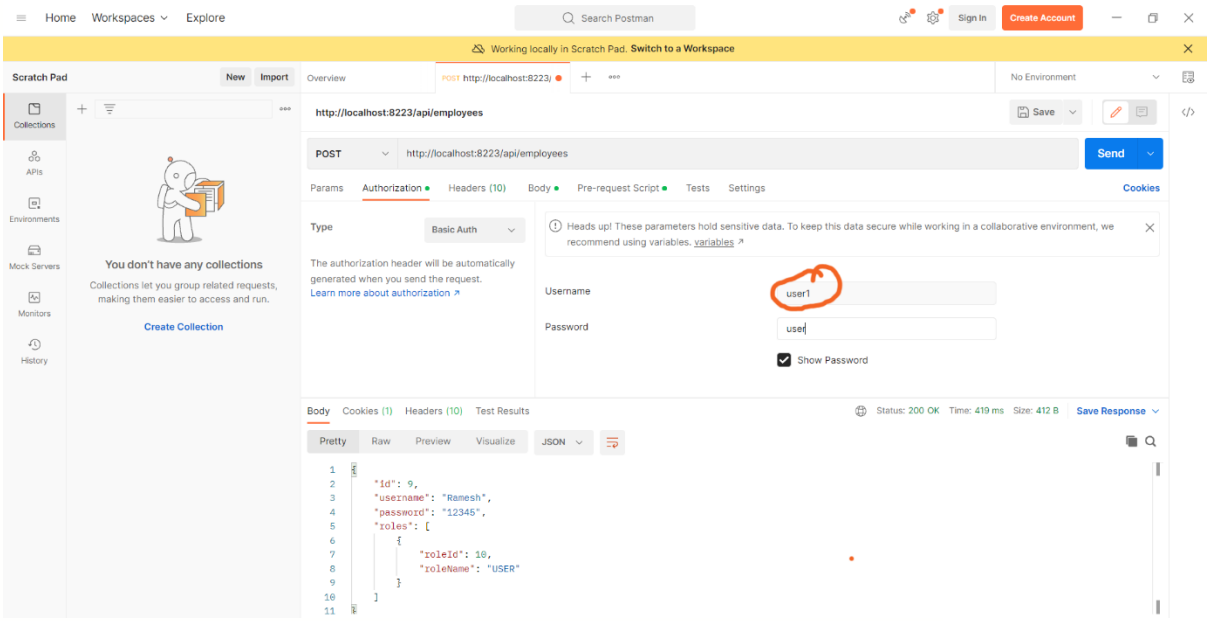


3. Now Your application should be able to add employees data in the db if and only if the authenticated user is **ADMIN**-

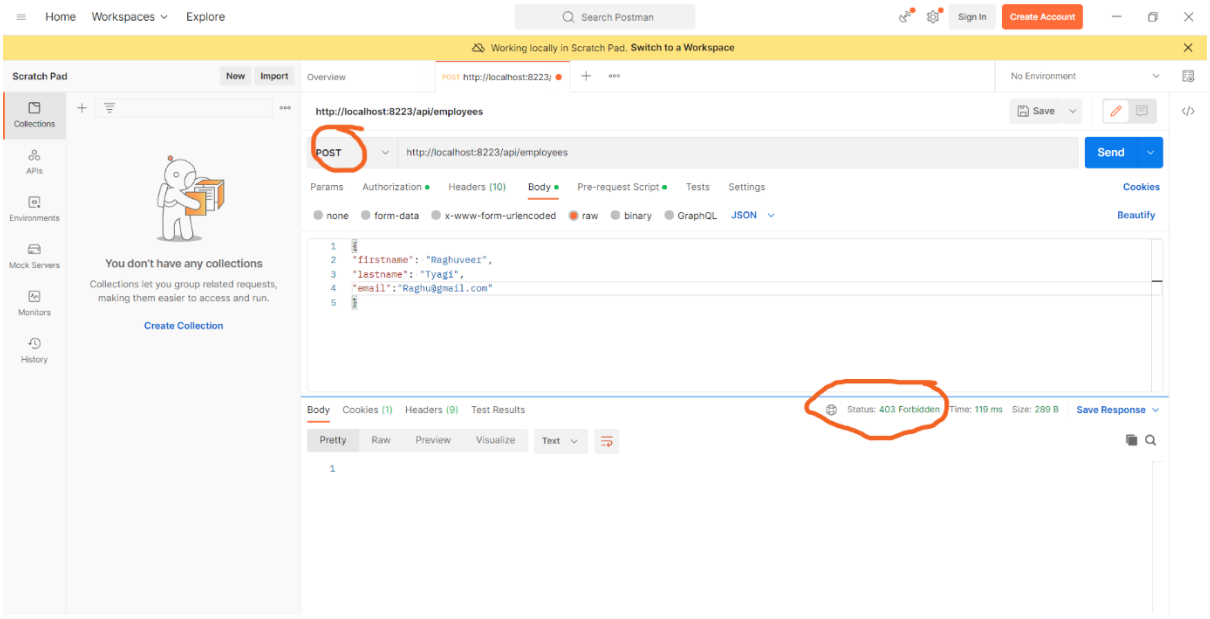
Ex-

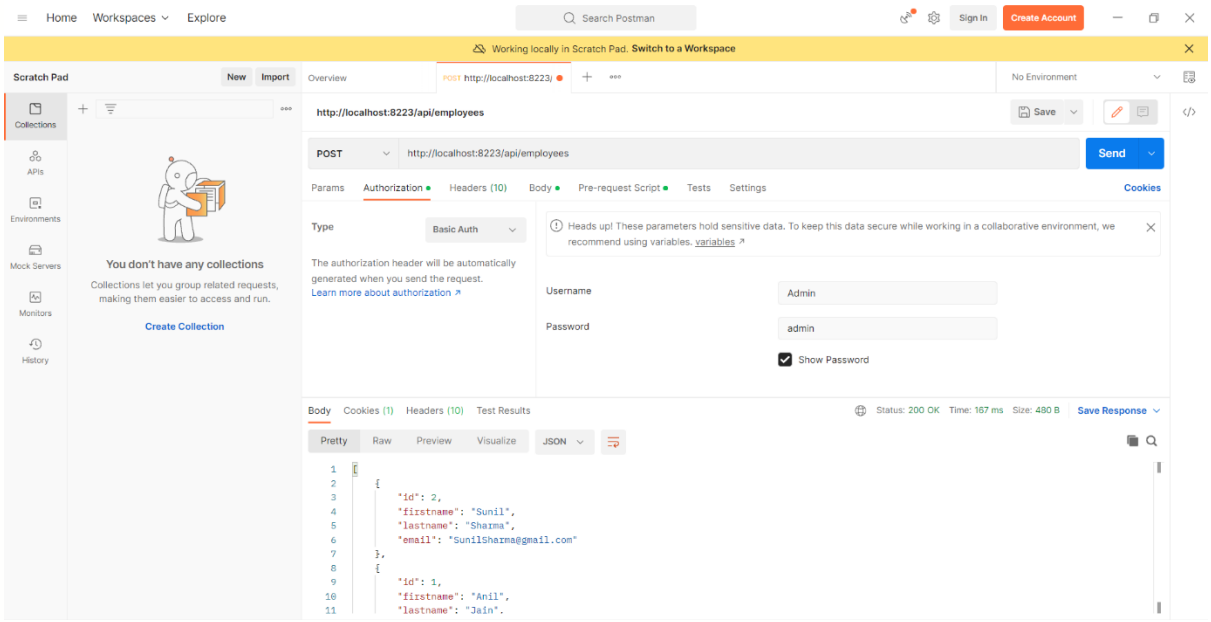
```
{
    "firstName":"gl",
    "lastName":"postman",
    "email":"postman@gamil.com"
```

}

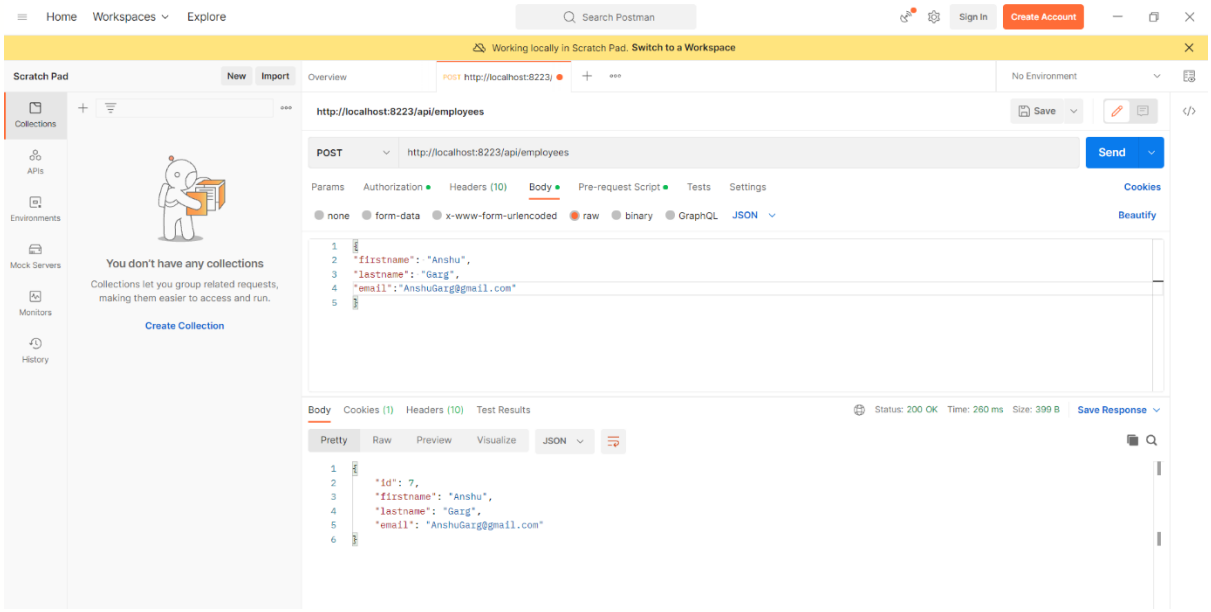


USER LOGGED IN AS A USER DOES NOT HAVE THE PERMISSION TO ADD EMPLOYEES.





USER LOGGED IN AS AN ADMIN CAN ADD EMPLOYEES



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

Ex-

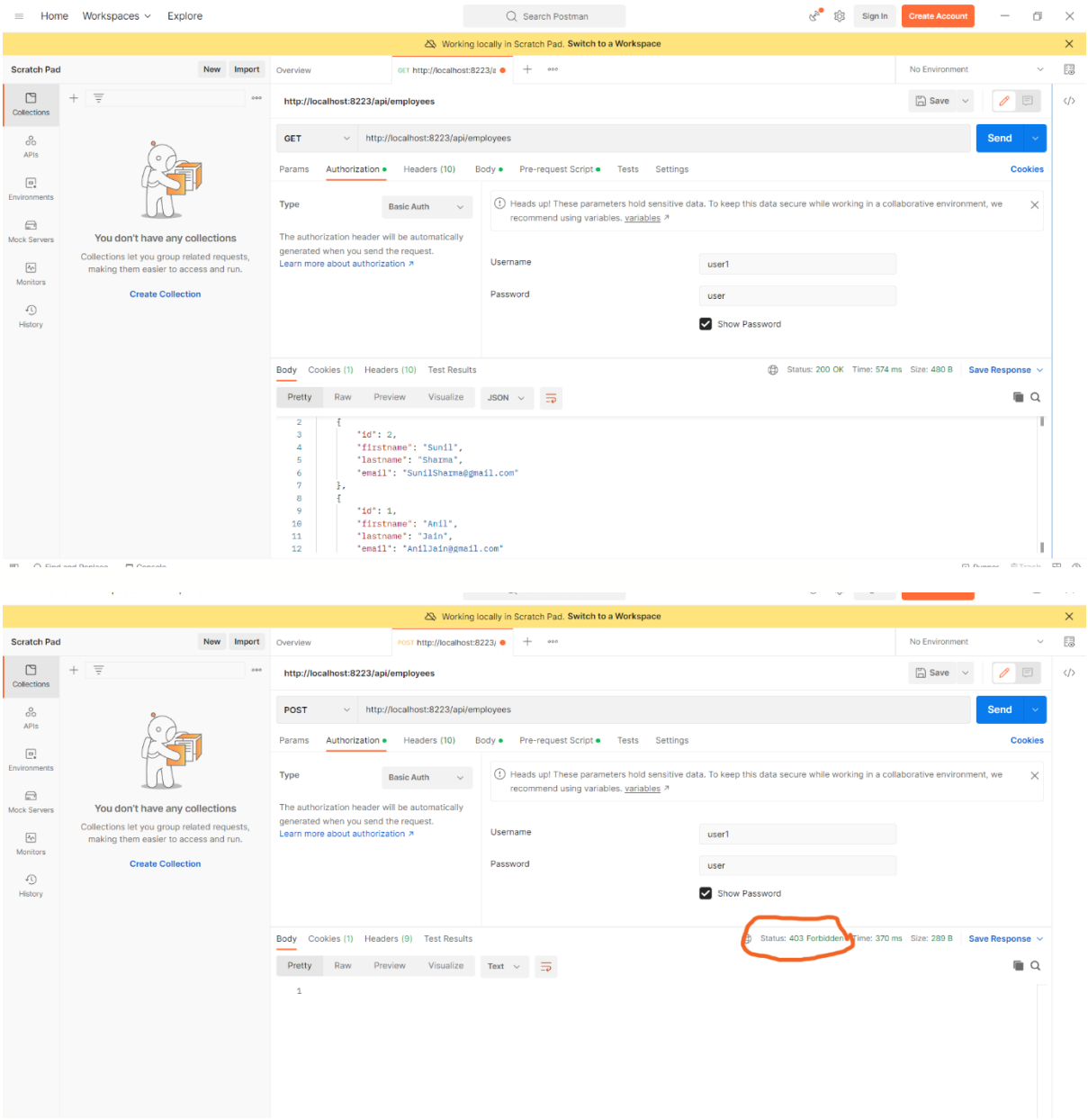
Response Body-

```
[
    {
        "id": 1,
```

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


USER LOGGED IN AS USER CAN GET THE EMPLOYEES FOR THE DATABASE.

USER LOGGED IN AS USER CANNOT ADD THE EMPLOYEES TO THE DATABASE.



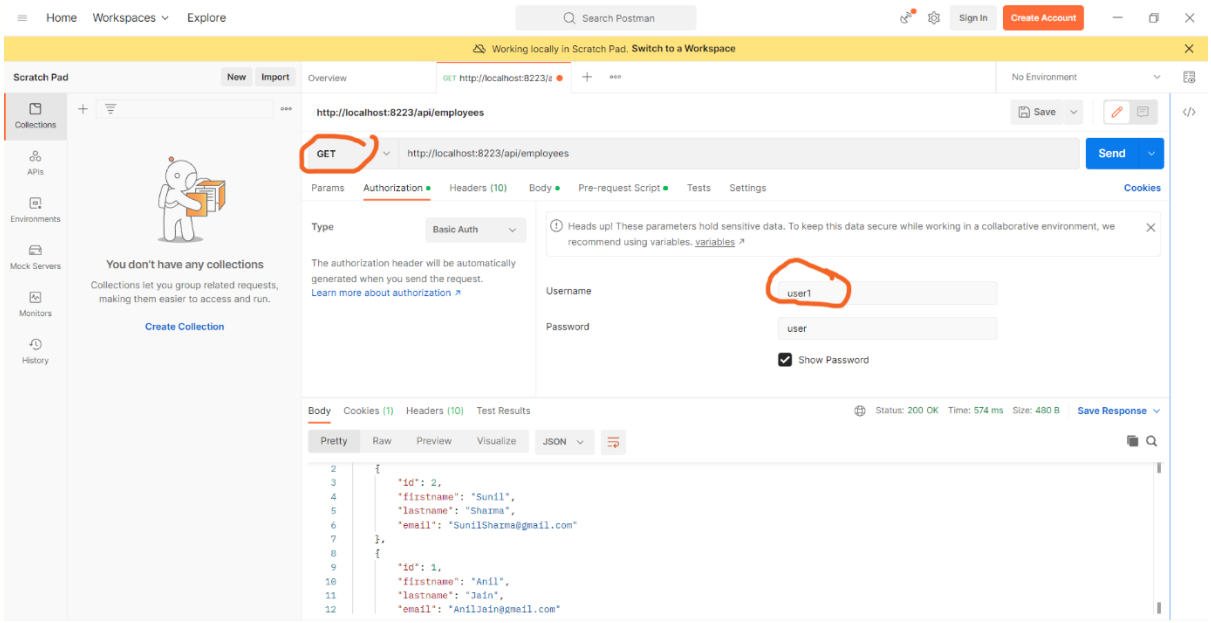
5. Your application should provide endpoint to fetch or get an employee record specifically based on the id of that employee-

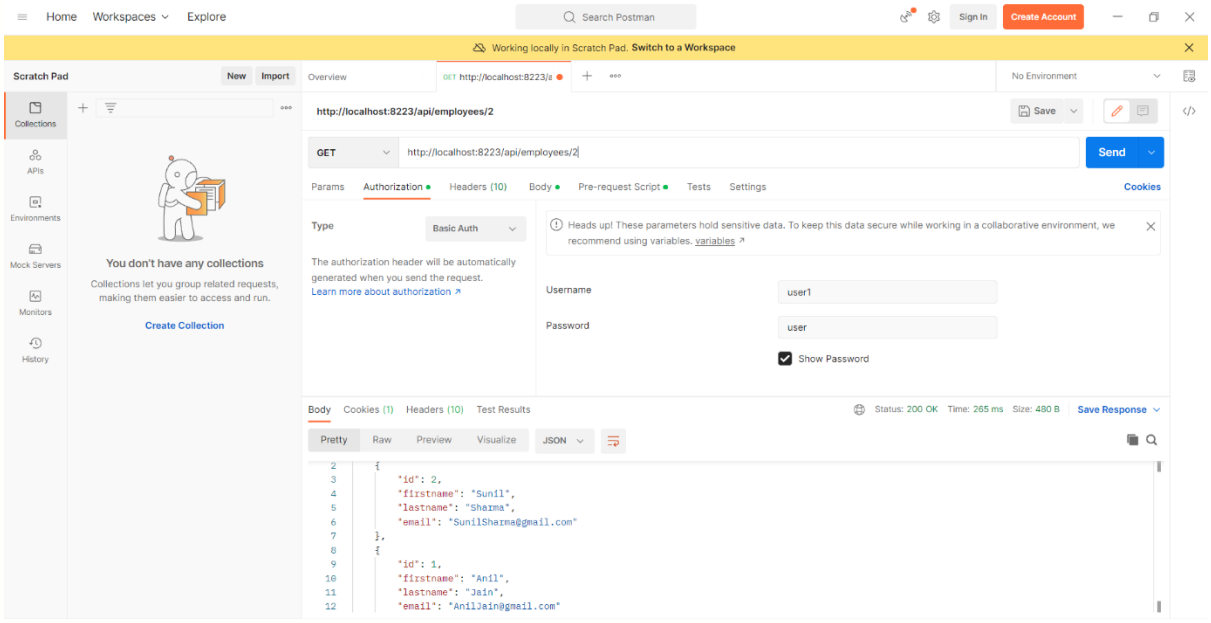
Ex- Url- http://localhost:8080/api/employees/3

Response Body-


```
{
    "id": 3,
    "firstName": "postman",
    "lastName": "postman",
```

```
    "email": "postman@gamil.com"
}
```





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

Ex-

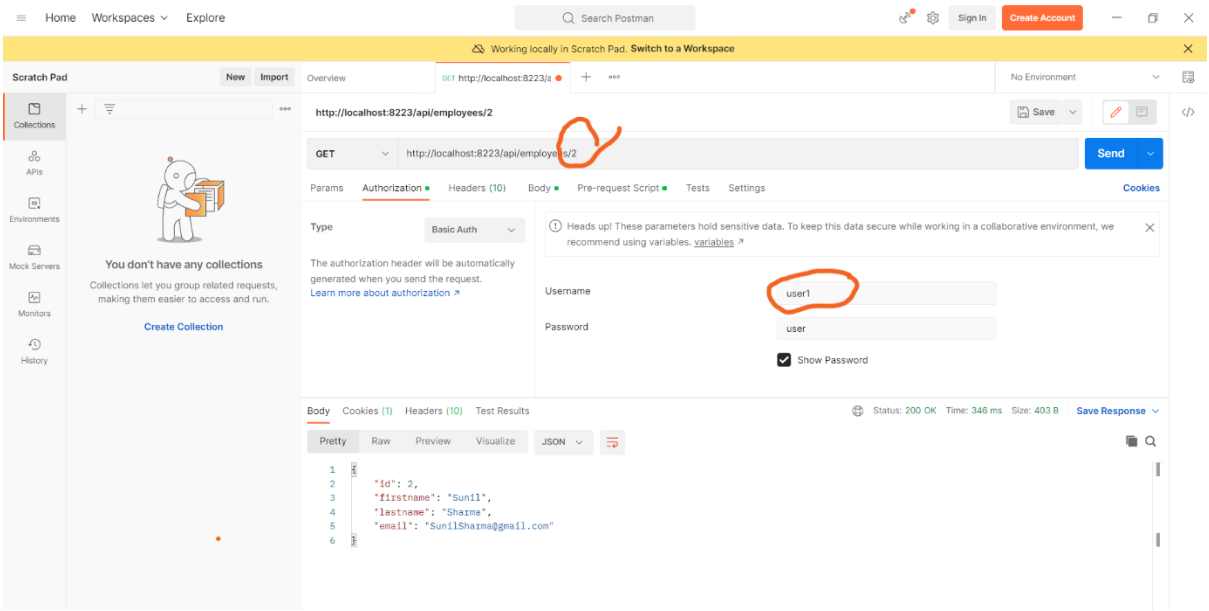
Object to be updated(raw->Json)-

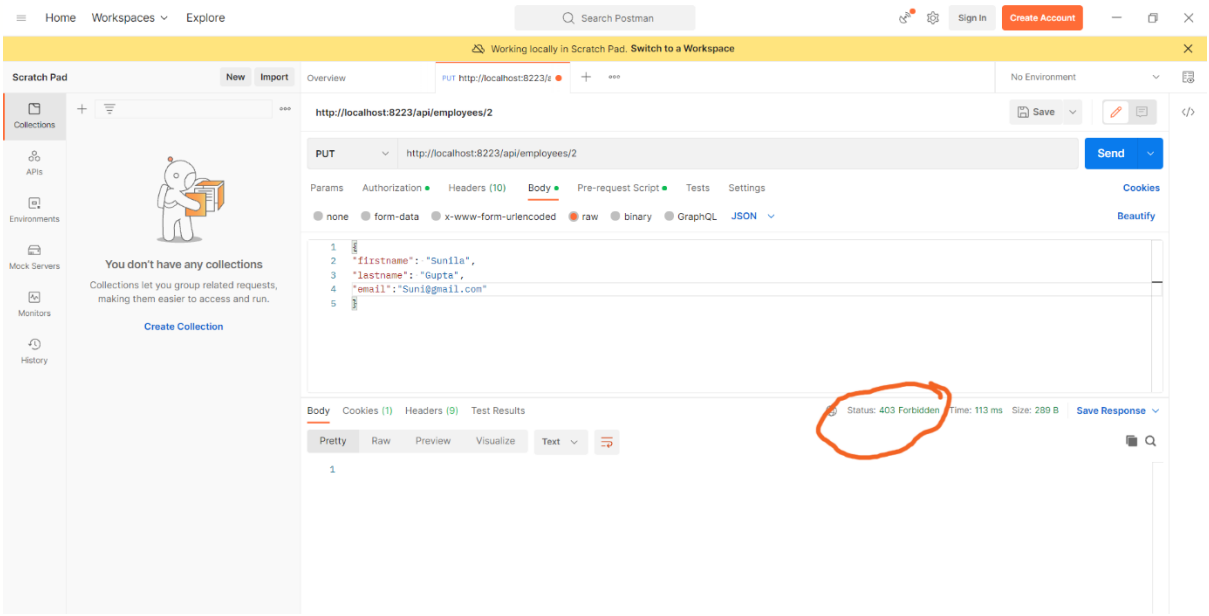
```
{
```

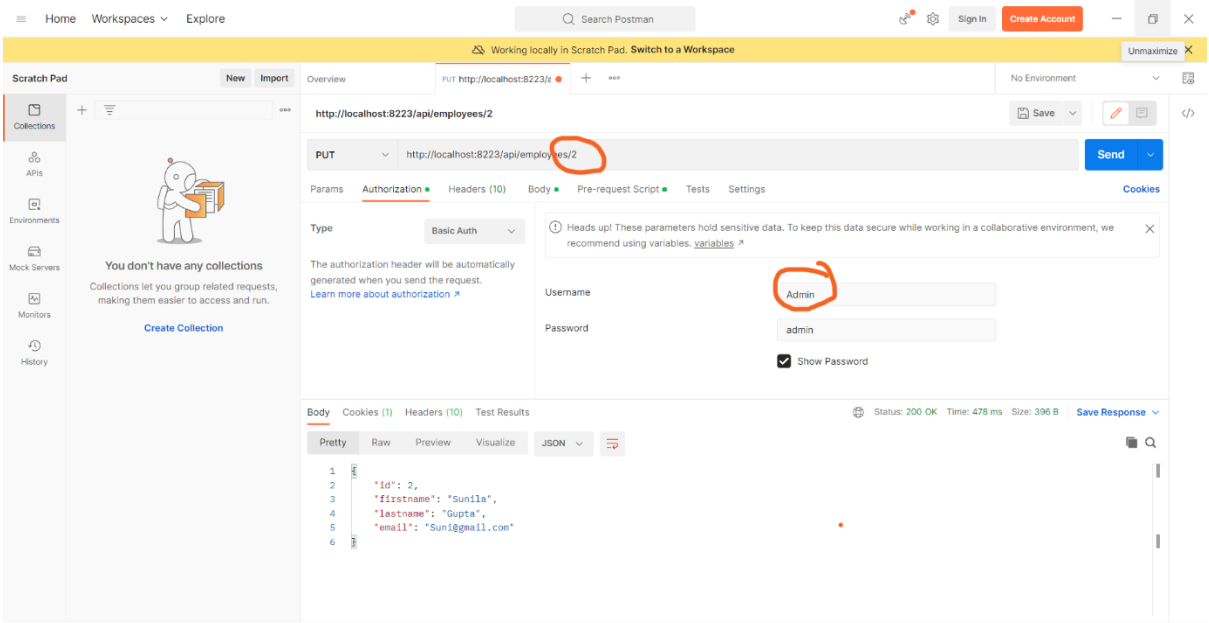
```
    "id":1,
    "firstName":"postman",
    "lastName":"postman",
    "email":"postman@gamil.com"
}
```

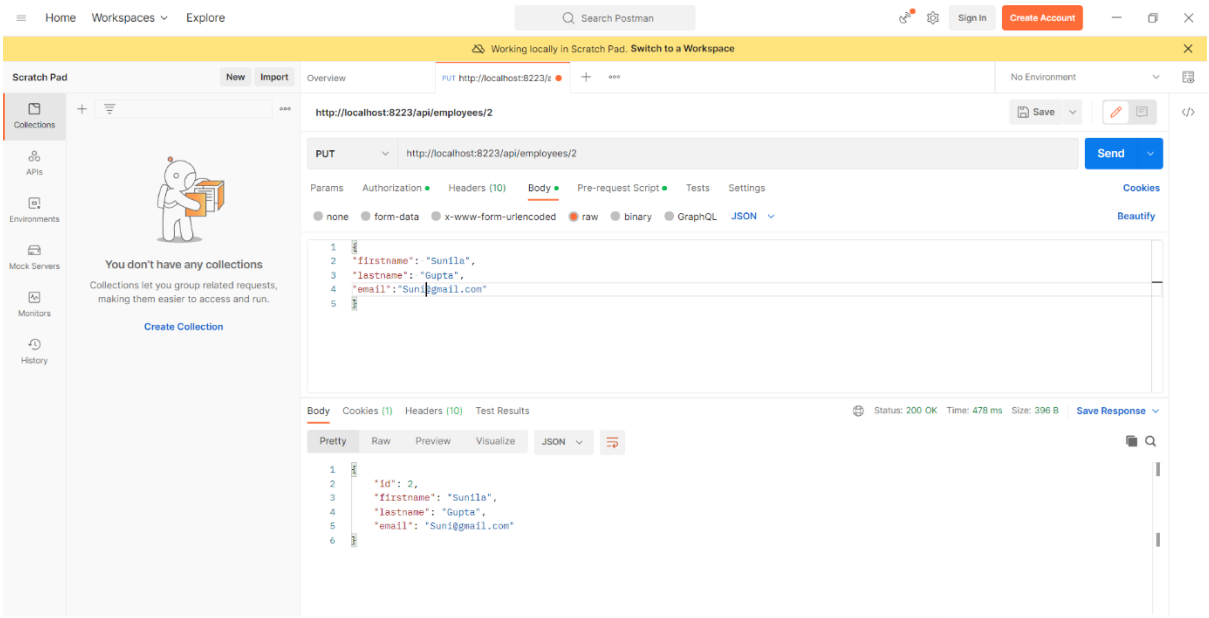

Response Body after updation-


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


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Overview
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
http://localhost:8223/api/employees/2
Save
PUT
http://localhost:8223/api/employees/2
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"firstname": "Sunila",
"lastname": "Gupta",
"email":"Suni@gmail.com"
}
Body
Cookies (1)
Headers (9)
Test Results
Status: 403 Forbidden
Time: 113 ms
Size: 289 B
Save Response
Pretty
Raw
Preview
Visualize
Text


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Unmaximize
Scratch Pad
New
Import
Overview
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
http://localhost:8223/api/employees/2
Save
PUT
http://localhost:8223/api/employees/2
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
Type
Basic Auth
The authorization header will be automatically generated when you send the request.
Learn more about authorization
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.
variables
Username
Admin
Password
admin
Show Password
Body
Cookies (1)
Headers (10)
Test Results
Status: 200 OK
Time: 478 ms
Size: 396 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 2,
"firstname": "Sunila",
"lastname": "Gupta",
"email": "Suni@gmail.com"
}



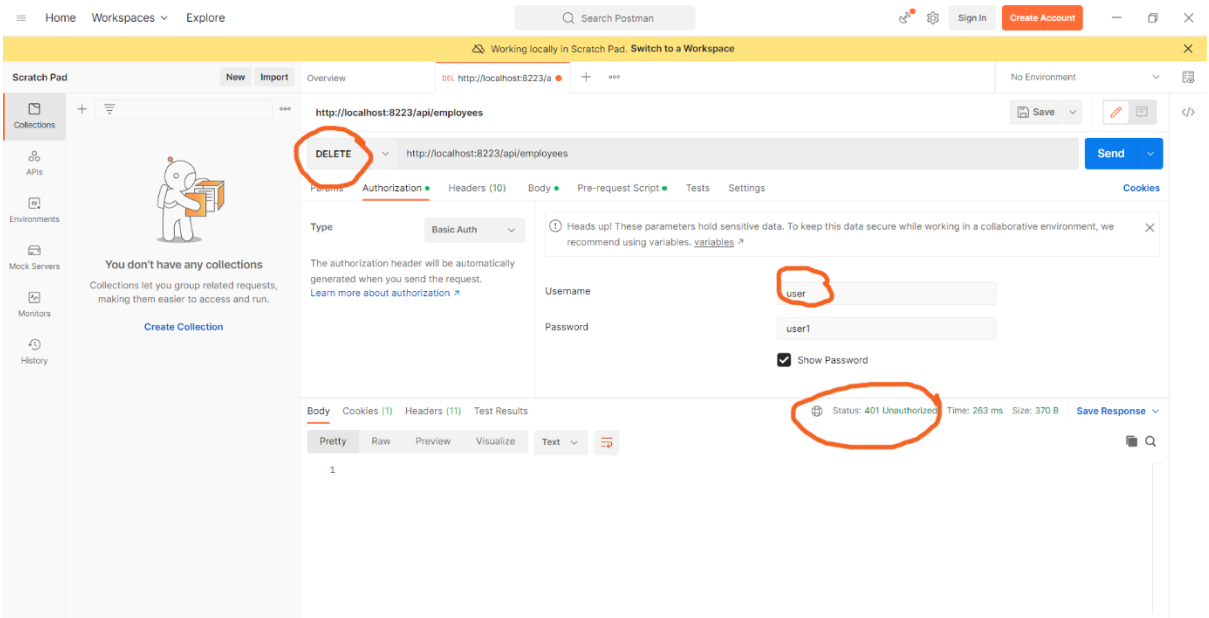
7. Your application should also provide an endpoint to delete an existing employee record based on the id of the employee-

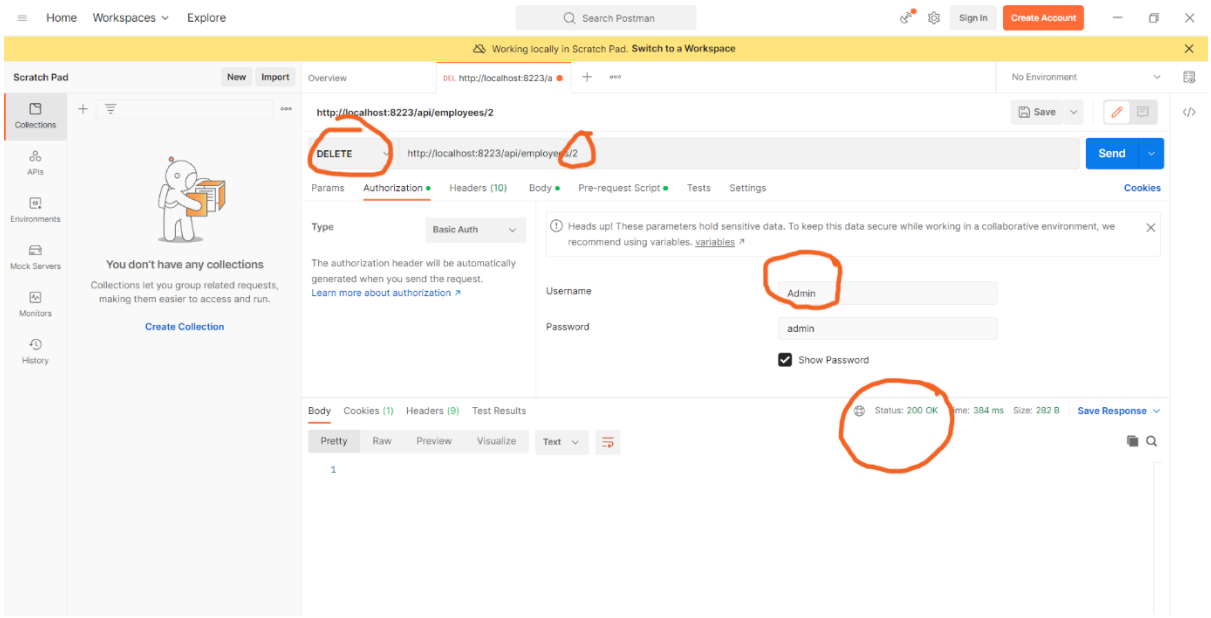
Ex-

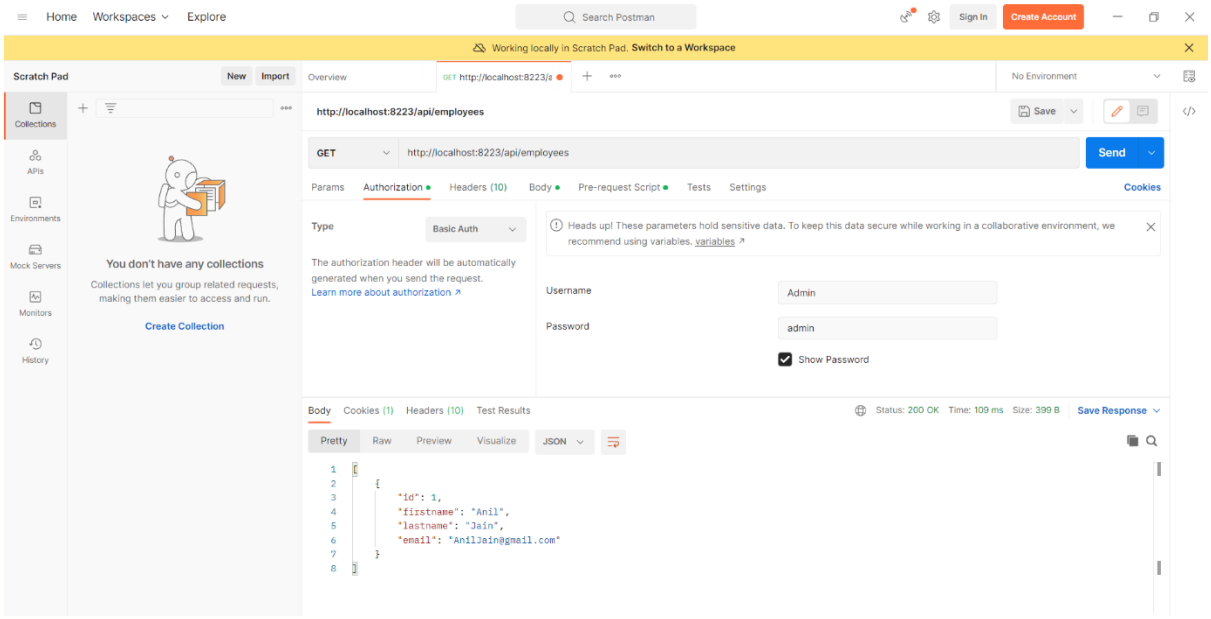
Url- http://localhost:8080/api/employees/4

Response Body-

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







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-

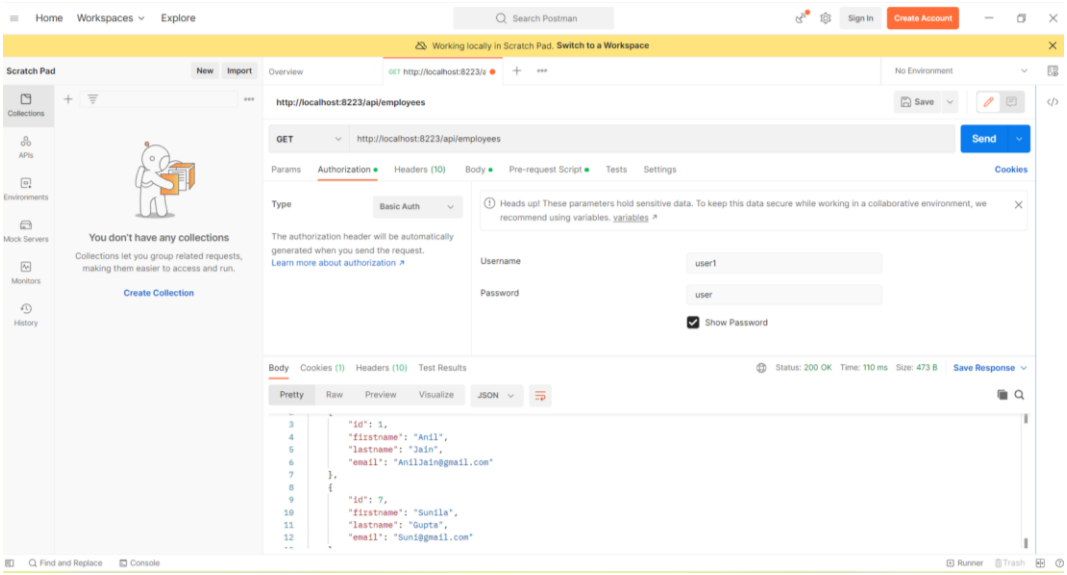
Ex-

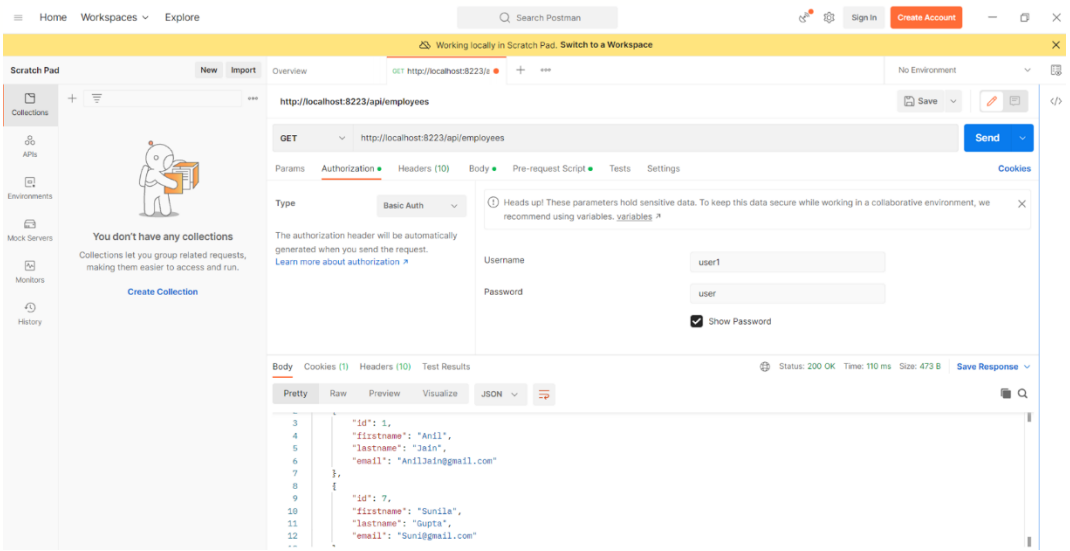
Url- http://localhost:8080/api/employees/search/gl

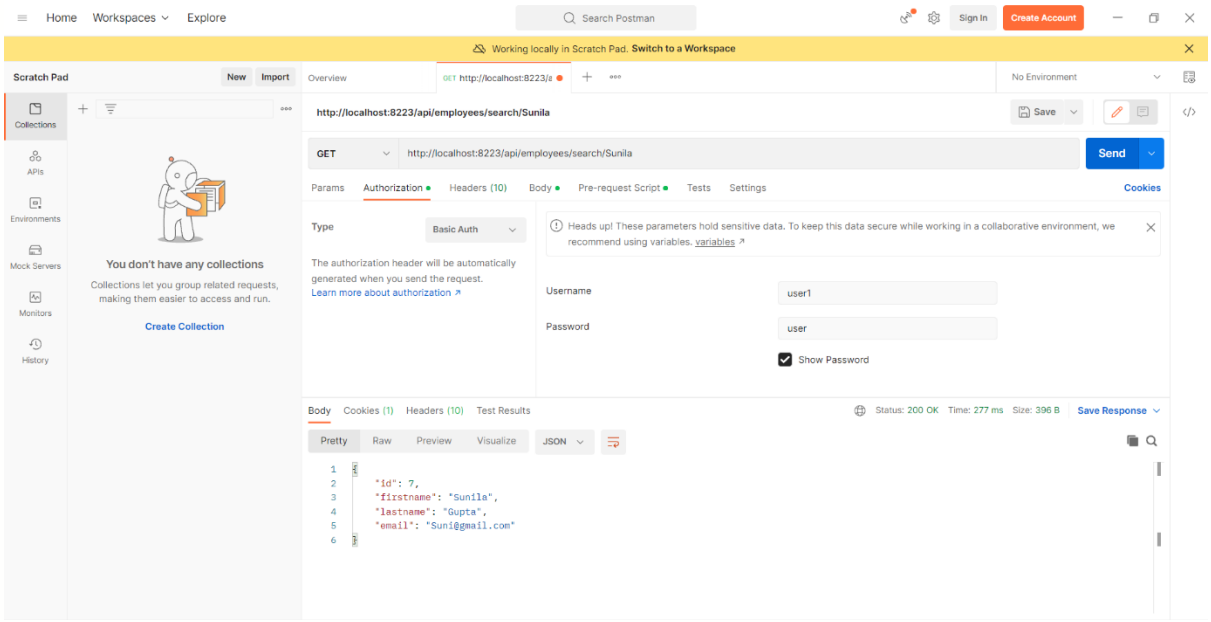
Response Body-

[

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





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

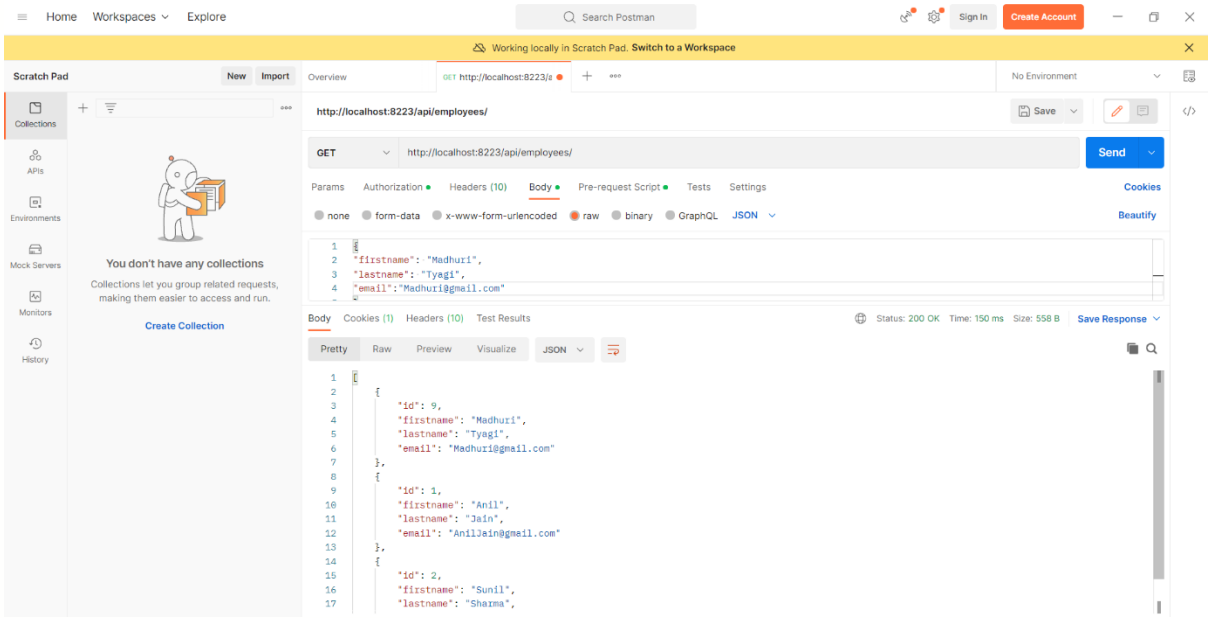
Ex-

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

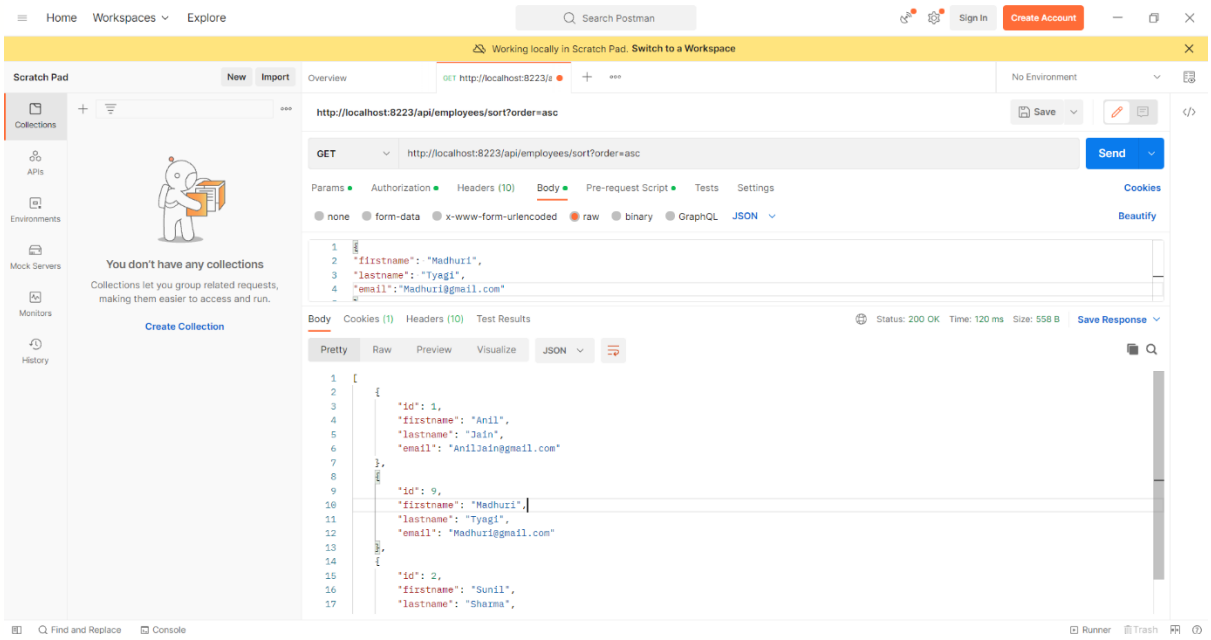
OR

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

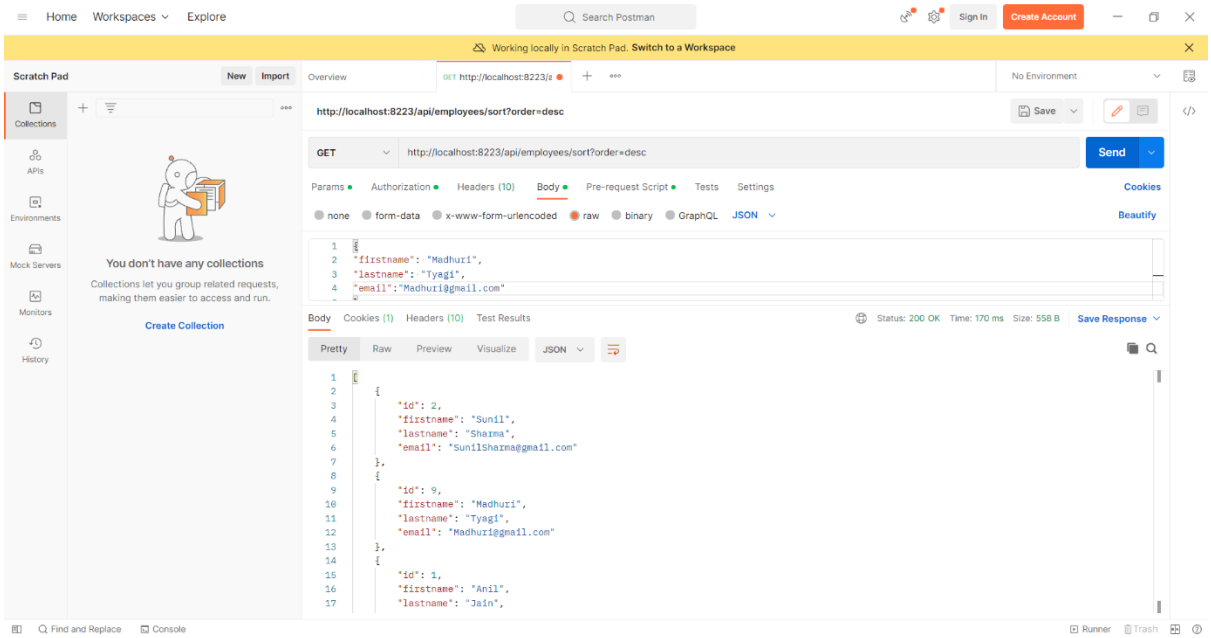
LIST OF EMPLOYEES –



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



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



Important instructions

i) You should use the H2 in Memory database/MySQL 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)