

Adding Role in database:

The screenshot shows the Postman application interface. On the left, the 'Scratch Pad' sidebar is visible with options for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main workspace displays a POST request to `http://localhost:8080/rolecontroller/role`. The request body is set to 'raw' and contains the following JSON:

```
1 {
2   "name": "ADMIN"
3 }
```

The response status is 200 OK, with a time of 619 ms and a size of 394 B. The response body is empty.

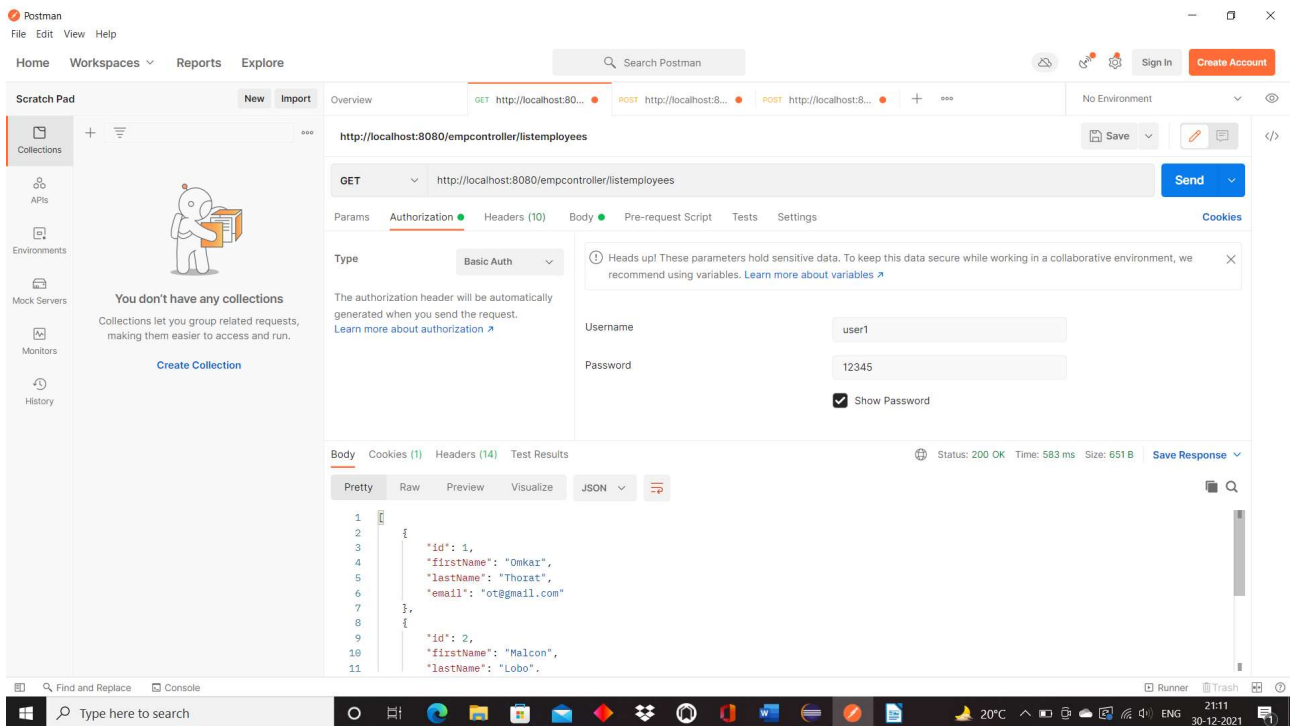
Adding User in database with BCrypt encrypted password (12345):

The screenshot shows the Postman application interface. On the left, the 'Scratch Pad' sidebar is visible with options for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main workspace displays a POST request to `http://localhost:8080/usercontroller/user`. The request body is set to 'raw' and contains the following JSON:

```
1 {
2   "username": "user1",
3   "password": "$2a$12$nlLS2SAvoPwM3ZA6tb1BKu31RLu3saN9esXYnVoSsKdoh6Xz/3v2y",
4   "roles": [
5     {
6       "name": "ADMIN"
7     }
8   ]
9 }
```

The response status is 200 OK, with a time of 99 ms and a size of 394 B. The response body is empty.

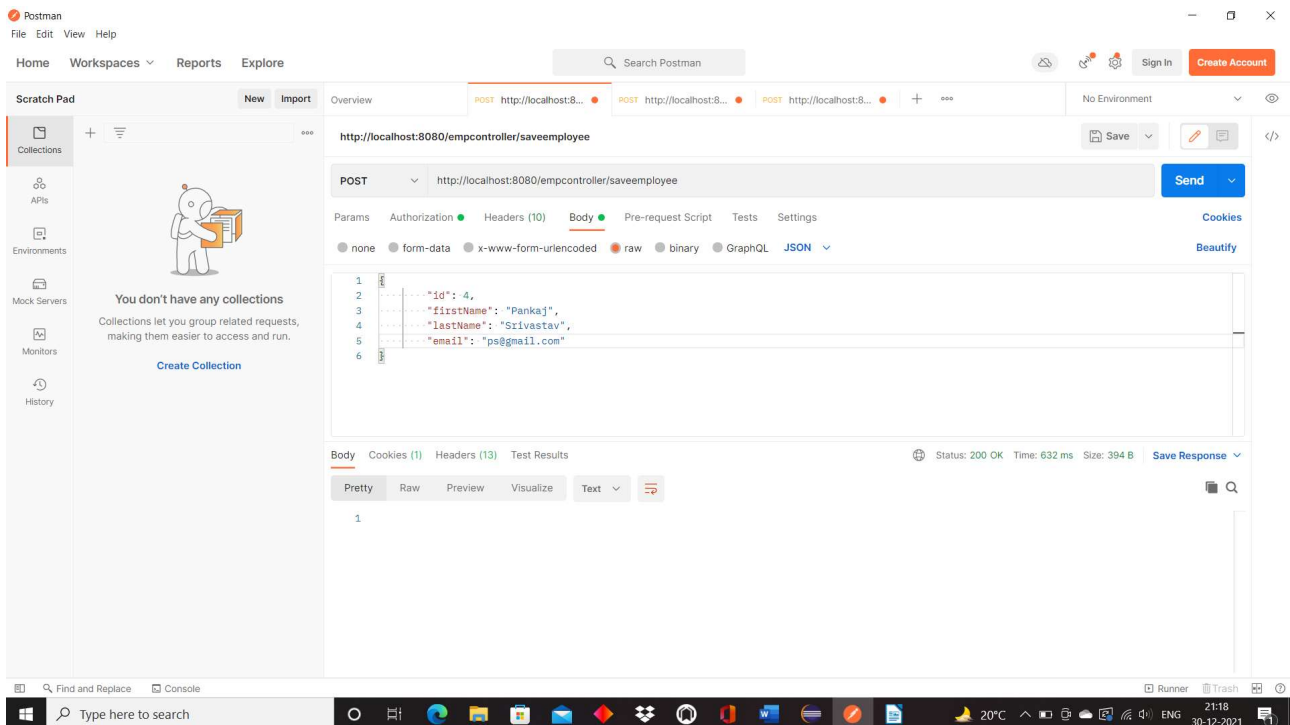
List all employees:



Postman interface showing a GET request to `http://localhost:8080/empcontroller/listemployees`. The request is configured with Basic Auth (Username: `user1`, Password: `12345`). The response is a JSON array of two employee objects.

```
1 {
2   "id": 1,
3   "firstName": "Omkar",
4   "lastName": "Thorat",
5   "email": "ot@gmail.com"
6 }
7 ,
8 {
9   "id": 2,
10  "firstName": "Malcon",
11  "lastName": "Lobo"
```

Add employee:



Postman interface showing a POST request to `http://localhost:8080/empcontroller/saveemployee`. The request is configured with a raw body containing a JSON object representing a new employee.

```
1 {
2   "id": 4,
3   "firstName": "Pankaj",
4   "lastName": "Srivastav",
5   "email": "ps@gmail.com"
6 }
```

Find employee by id:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections', 'Environments', 'Mock Servers', 'Monitors', and 'History'. The main area displays a GET request to `http://localhost:8080/empcontroller/findemployee?id=3`. The request is configured with Basic Auth, using 'user1' as the username and '12345' as the password. The response is shown in the 'Body' tab, displaying a JSON object:

```
{  "id": 3,  "firstName": "Ayush",  "lastName": "Sharma",  "email": "as@gmail.com"}
```

. The status is 200 OK, with a time of 645 ms and a size of 506 B.

Updating employee details (updated email):

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections', 'Environments', 'Mock Servers', 'Monitors', and 'History'. The main area displays a POST request to `http://localhost:8080/empcontroller/saveemployee`. The request is configured with 'raw' body type. The body contains a JSON object:

```
{  "id": 3,  "firstName": "Ayush",  "lastName": "Sharma",  "email": "as@greatlearning.in"}
```

. The response is shown in the 'Body' tab, displaying a single line of text:

```
1
```

. The status is 200 OK, with a time of 605 ms and a size of 394 B.

Delete employee by id:

The screenshot shows the Postman application interface. On the left, the 'Scratch Pad' sidebar is visible with a message: 'You don't have any collections. Collections let you group related requests, making them easier to access and run. Create Collection'. The main workspace displays a DELETE request to the URL `http://localhost:8080/empcontroller/deleteemployee?id=4`. The request is configured with Basic Authentication using the username 'user1' and password '12345'. The 'Show Password' checkbox is checked. The 'Body' tab is selected, showing a status of 200 OK, a time of 537 ms, and a size of 394 B. The response is displayed in the 'Body' tab, showing a single line of text: '1'.

Fetch employee by first name:

The screenshot shows the Postman application interface. On the left, the 'Scratch Pad' sidebar is visible with a message: 'You don't have any collections. Collections let you group related requests, making them easier to access and run. Create Collection'. The main workspace displays a GET request to the URL `http://localhost:8080/empcontroller/employeewithname?firstName=Malcon`. The request is configured with Basic Authentication using the username 'user1' and password '12345'. The 'Show Password' checkbox is checked. The 'Body' tab is selected, showing a status of 200 OK, a time of 507 ms, and a size of 507 B. The response is displayed in the 'Body' tab, showing a JSON object with employee details:

```
1 {
2   "id": 2,
3   "firstName": "Malcon",
4   "lastName": "Lobo",
5   "email": "m1@gmail.com"
6 }
7
8
```

List all employees with ascending/descending order of first names:

Postman interface showing a GET request to `http://localhost:8080/empcontroller/sort?direction=ASC` with Basic Auth credentials (user1, 12345). The response is a JSON array of employee objects:

```
1 {
2   "id": 3,
3   "firstName": "Ayush",
4   "lastName": "Sharma",
5   "email": "as@greatlearning.in"
6 },
7 {
8   "id": 2,
9   "firstName": "Malcon",
10  "lastName": "Lobo"
11 }
```

Database after all operations:

H2 Console interface showing the SQL statement `SELECT * FROM EMPLOYEE;` and the resulting table:

EMPLOYEE_ID	EMAIL	FIRST_NAME	LAST_NAME
1	ot@gmail.com	Omkar	Thorat
2	mtl@gmail.com	Malcon	Lobo
3	as@greatlearning.in	Ayush	Sharma

(3 rows, 6 ms)