

# USER-

## 1)Create New User

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar lists collections, including "User Provisioning" which contains a "User" folder. The "User" folder is expanded, showing a list of requests: "POST Create User", "GET Get all users", "GET Get user by Id", "GET Get User by Page", "GET Search by name and Email", "PUT update user status", "DEL Change user status", and "DEL User Delete". The "POST Create User" request is selected. The main panel shows the request details: Method "POST", URL "http://localhost:8080/users", and a JSON body with the following content:

```
{  "name": "mohit",  "email": "mht4943@gmail.com"}
```

The response is displayed below the request, showing a status of "201 Created", a response time of "495 ms", and a body of "1 User Created Successfully".

## 2)Get All Users

The screenshot shows the Postman interface with the same workspace. The "GET Get all users" request is selected. The main panel shows the request details: Method "GET", URL "http://localhost:8080/users", and no body. The response is displayed below the request, showing a status of "200 OK", a response time of "40 ms", and a JSON body with the following content:

```
[  {    "id": 1,    "name": "mohit",    "email": "mht4943@gmail.com",    "active": true,    "registrationDate": "2025-01-23T07:11:16.744+00:00"  },  {    "id": 2,    "name": "Nidhi",    "email": "Nidho@gmail.com",    "active": true,    "registrationDate": "2025-01-23T07:14:47.951+00:00"  }]
```

### 3)Get user by Id

The screenshot shows the Postman interface with a collection named 'User Provisioning' expanded to the 'User' folder. The selected endpoint is 'GET Get user by Id'. The request is a GET method to 'http://localhost:8080/users/2'. The response is a 200 OK status with a response time of 24 ms and a body size of 278 B. The JSON response is as follows:

```
1 {
2   "id": 2,
3   "name": "Nidhi",
4   "email": "Nidho@gmail.com",
5   "active": true,
6   "registrationDate": "2025-01-23T07:14:47.951+00:00"
7 }
```

### 4)Get user by page

The screenshot shows the Postman interface with the same 'User Provisioning' collection. The selected endpoint is 'GET Get User by Page'. The request is a GET method to 'http://localhost:8080/users/paginated?page=0&size=1'. The response is a 200 OK status with a response time of 46 ms and a body size of 595 B. The JSON response is a paginated list of users:

```
1 {
2   "content": [
3     {
4       "id": 1,
5       "name": "mohit",
6       "email": "mht4943@gmail.com",
7       "active": true,
8       "registrationDate": "2025-01-23T07:11:16.744+00:00"
9     }
10  ],
11  "pageable": {
12    "pageNumber": 0,
13    "pageSize": 1,
14    "sort": {
15      "empty": true,
16      "sorted": false,
17      "unsorted": true
18    },
19    "offset": 0,
20    "paged": true,
21    "unpaged": false
22  },
23  "last": false,
24  "totalElements": 2,
25  "totalPages": 2,
26  "first": true,
```

## 5)Get by name or email

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar lists collections, environments, flows, and history. The "User" collection is expanded, showing endpoints like "POST Create User", "GET Get all users", "GET Get user by Id", "GET Get User by Page", "GET Search by name and Email", "PUT update user status", "DEL Change user status", and "DEL User Delete". The "GET Search by name and Email" endpoint is selected.

The main panel shows the request details for the "GET Search by name and Email" endpoint. The URL is `http://localhost:8080/users/search?name=n...`. The method is "GET". The "Params" tab is active, showing a table of query parameters:

Key	Value	Description
name	mohit	
email		

The "Body" tab shows the response, which is a 200 OK status with a response time of 29 ms and a body size of 282 B. The response is in JSON format:

```
{  "id": 1,  "name": "mohit",  "email": "mht4943@gmail.com",  "active": true,  "registrationDate": "2025-01-23T07:11:16.744+00:00"}
```

## 6)Update User Status

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar lists collections, environments, flows, and history. The "User" collection is expanded, showing endpoints like "POST Create User", "GET Get all users", "GET Get user by Id", "GET Get User by Page", "GET Search by name and Email", "PUT update user status", "DEL Change user status", and "DEL User Delete". The "PUT update user status" endpoint is selected.

The main panel shows the request details for the "PUT update user status" endpoint. The URL is `http://localhost:8080/users/1`. The method is "PUT". The "Body" tab is active, showing the request body in JSON format:

```
{  "name": "mohitUpdated",  "email": "mht4943@gmail.com",  "active": false}
```

The "Body" tab shows the response, which is a 200 OK status with a response time of 155 ms and a body size of 288 B. The response is in JSON format:

```
{  "id": 1,  "name": "mohitUpdated",  "email": "mht4943@gmail.com",  "active": false,  "registrationDate": "2025-01-23T07:11:16.744+00:00"}
```

## 7)Change User Status

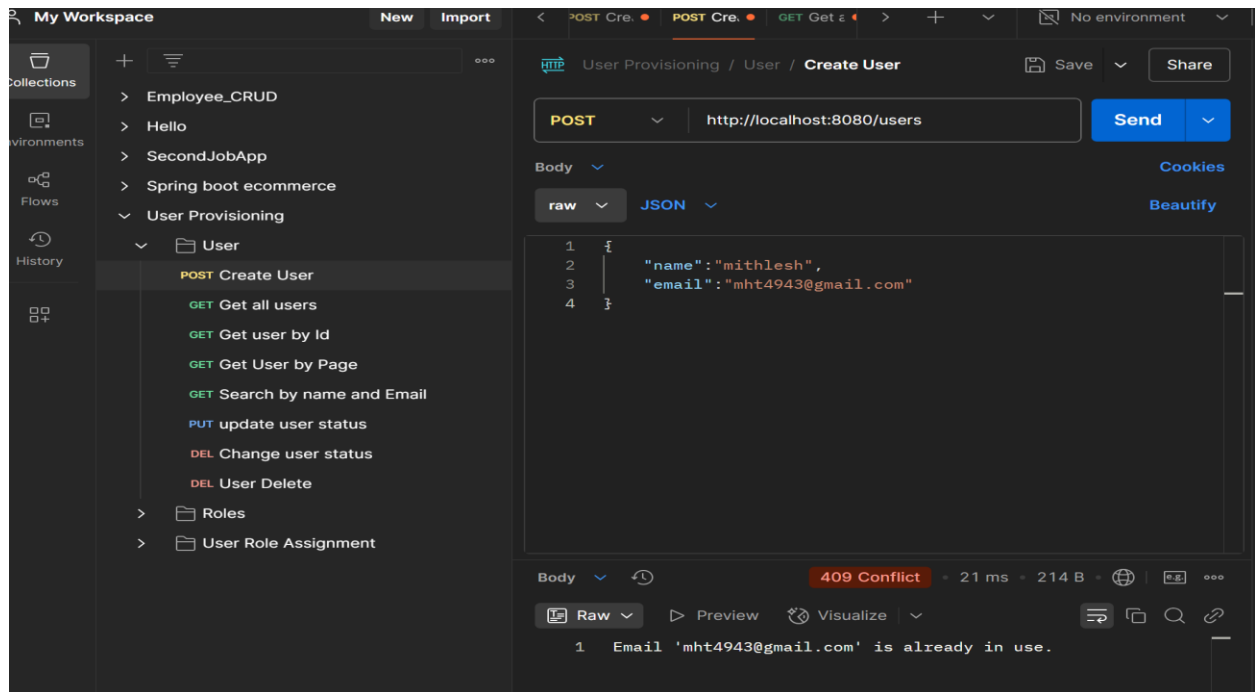
The screenshot shows a REST client interface with a sidebar on the left containing 'My Workspace', 'Collections', 'Environments', 'Flows', and 'History'. The 'Collections' section is expanded, showing a hierarchy: 'Employee\_CRUD' > 'Hello' > 'SecondJobApp' > 'Spring boot ecommerce' > 'User Provisioning' > 'User'. The 'User' collection is selected, and the 'DEL Change user status' endpoint is highlighted.

The main panel displays the 'Change user status' endpoint with a 'DELETE' method and the URL 'http://localhost:8080/users/2'. The 'Send' button is visible. Below the URL bar, there are sections for 'Params' and 'Query Params'. The 'Body' section shows a status of '200 OK' with a response time of 29 ms and a body size of 184 B. The response body is displayed in 'Raw' format as '1 Changed successfully'.

## 8)Delete User

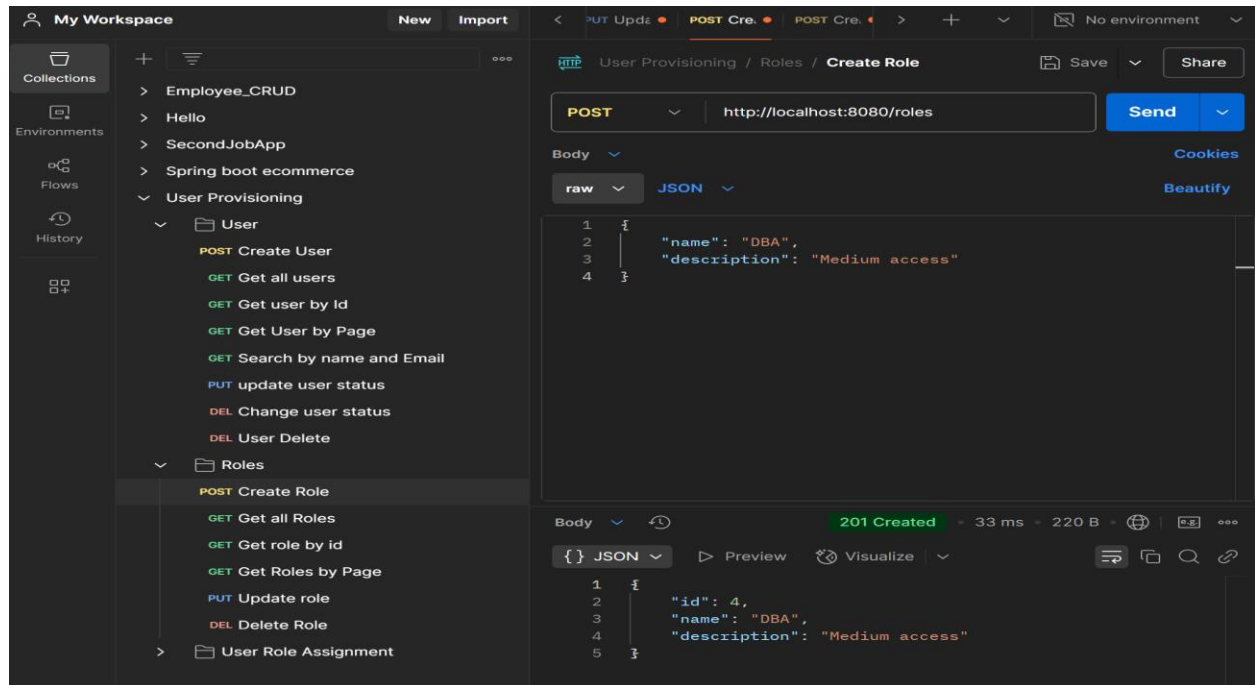
The screenshot shows the same REST client interface as the previous one, but with the 'DEL User Delete' endpoint selected in the 'User' collection. The main panel displays the 'User Delete' endpoint with a 'DELETE' method and the URL 'http://localhost:8080/users/delete/2'. The 'Send' button is visible. Below the URL bar, there are sections for 'Params' and 'Query Params'. The 'Body' section shows a status of '200 OK' with a response time of 35 ms and a body size of 184 B. The response body is displayed in 'Raw' format as '1 Deleted successfully'.

## 9) Duplicate Email



## ROLES-

### 1) Create Role



## 2)Get all roles

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar displays a collection of API requests under "User Provisioning" > "Roles". The "Get all Roles" request is selected. The main panel shows the request details for a GET request to `http://localhost:8080/roles`. The response is a 200 OK status with a response time of 10 ms and a body size of 325 B. The response body is displayed in JSON format, showing an array of three role objects:

```
[{"id": 1, "name": "Admin", "description": "Strong access"}, {"id": 2, "name": "Manager", "description": "Limited access"}, {"id": 3, "name": "User", "description": "No access"}]
```

## 3)Role already exist

The screenshot shows the Postman interface with the same workspace. The "Create Role" request is selected, which is a POST request to `http://localhost:8080/roles`. The request body is a JSON object: `{"name": "User", "description": "No access"}`. The response is a 409 Conflict status with a response time of 8 ms and a body size of 207 B. The response body is displayed in raw text format, showing the error message: "Role with name 'User' already exists."

#### 4)Get role by Id

The screenshot shows the Postman interface with a workspace named 'My Workspace'. The left sidebar lists collections, including 'User Provisioning' which contains a 'Roles' folder. The 'Roles' folder is expanded, showing a list of endpoints. The 'GET Get role by id' endpoint is selected. The main panel shows the request details: Method 'GET', URL 'http://localhost:8080/roles/3', and a status of '200 OK' with a response time of 11 ms and a body size of 212 B. The response body is displayed in JSON format:

```
{  "id": 3,  "name": "User",  "description": "No access"}
```

#### 5)Get roles by Page

The screenshot shows the Postman interface with a workspace named 'My Workspace'. The left sidebar lists collections, including 'User Provisioning' which contains a 'Roles' folder. The 'Roles' folder is expanded, showing a list of endpoints. The 'GET Get Roles by Page' endpoint is selected. The main panel shows the request details: Method 'GET', URL 'http://localhost:8080/roles/paginated?page=...', and a status of '200 OK' with a response time of 23 ms and a body size of 589 B. The response body is displayed in JSON format:

```
{  "id": 1,  "name": "Admin",  "description": "Strong access"},  {  "id": 2,  "name": "Manager",  "description": "Limited access"}],  "pageable": {    "pageNumber": 0,    "pageSize": 2,    "sort": {      "empty": true,      "sorted": false,      "unsorted": true    },    "offset": 0,    "paged": true,    "unpaged": false  },  "last": false,  "totalElements": 3,  "totalPages": 2,  "first": true,  "size": 2,  "number": 0,  "sort": {    "empty": true,    "sorted": false,    "unsorted": true  },  "numberOfElements": 2,  "empty": false}
```

## 6)Update Role

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar displays a collection of API requests under "User Provisioning" > "Roles". The selected request is "PUT Update role". The main panel shows the request details:

- Method: PUT
- URL: `http://localhost:8080/roles/1`
- Body: JSON
- Raw body content:

```
1 {
2   "name": "Super Admin",
3   "description": "extended access"
4 }
```
- Response: 200 OK, 26 ms, 225 B
- Response body (JSON):

```
1 {
2   "id": 1,
3   "name": "Super Admin",
4   "description": "extended access"
5 }
```

## 7)Delete role by Id

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar displays a collection of API requests under "User Provisioning" > "Roles". The selected request is "DEL Delete Role". The main panel shows the request details:

- Method: DELETE
- URL: `http://localhost:8080/roles/4`
- Params: None
- Query Params: None
- Response: 204 No Content, 26 ms, 112 B
- Response body: 1



# USER\_ROLE\_ASSIGNMENT

## 1) Assign new Role

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar lists collections, environments, flows, and history. The "User Provisioning" collection is expanded, showing a folder "User" with several endpoints. The "Assign New Role" endpoint is selected, showing a POST request to `http://localhost:8080/user-roles?userId=3&r...`. The response is a 201 Created status with a 29 ms response time and 412 B of data. The response body is a JSON object:

```
{  "id": 6,  "user": {    "id": 3,    "name": "mithlesh",    "email": "mithlesh@gmail.com",    "active": true,    "registrationDate": "2025-01-23T07:45:17.599+00:00"  },  "role": {    "id": 3,    "name": "User",    "description": "No access"  },  "registrationDate": "2025-01-23T07:45:52.953+00:00"}
```

## 2) Duplicate role Assignment for particular user

The screenshot shows the Postman interface with the same workspace. The "Assign New Role" endpoint is selected, but the URL is `http://localhost:8080/user-roles?userId=1&r...`. The response is a 409 Conflict status with a 17 ms response time and 205 B of data. The response body is a raw text message:

```
1 User is already assigned this role.
```

### 3)Get all user roles

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/user-roles?userId=&roleId=`. The response is a 200 OK status with a JSON body. The JSON body contains an array of objects, each representing a user and their role. The first object has a user with id 3, name 'mithlesh', email 'mithlesh@gmail.com', active status, and a registration date of '2025-01-23T07:45:17.599+00:00'. The role for this user is 'User' with id 3, name 'User', description 'No access', and a registration date of '2025-01-23T07:45:52.953+00:00'. The second object has a user with id 7, name 'mohitUpdated', email 'mht4943@gmail.com', active status, and a registration date of '2025-01-23T07:11:16.744+00:00'. The role for this user is 'Manager' with id 2, name 'Manager', description 'Limited access', and a registration date of '2025-01-23T07:49:18.800+00:00'.

```
4  "user": {
5    "id": 3,
6    "name": "mithlesh",
7    "email": "mithlesh@gmail.com",
8    "active": true,
9    "registrationDate": "2025-01-23T07:45:17.599+00:00"
10  },
11  "role": {
12    "id": 3,
13    "name": "User",
14    "description": "No access"
15  },
16  "registrationDate": "2025-01-23T07:45:52.953+00:00"
17  },
18  {
19    "id": 7,
20    "user": {
21      "id": 1,
22      "name": "mohitUpdated",
23      "email": "mht4943@gmail.com",
24      "active": true,
25      "registrationDate": "2025-01-23T07:11:16.744+00:00"
26    },
27    "role": {
28      "id": 2,
29      "name": "Manager",
30      "description": "Limited access"
31    },
32    "registrationDate": "2025-01-23T07:49:18.800+00:00"
33  }
```

### 4)Delete role by UserId

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8080/user-roles/1`. The response is a 204 No Content status. The interface shows the 'Query Params' section with a table containing two rows: 'Key' and 'Value'.

Key	Value	Des...	Bulk Edit
Key	Value	Description	