

Question-

You are required to create an 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-

- Get Role Details
- Add a new Role
- Get User Details
- Add a new User
- Save a new Employee
- Read Employee List
- Fetch Employee Data BY Id
- Update an existing Employee
- Delete an existing Employee
- Search Employee BY First Name
- Sort Employee BY First Name IN ASC / DESC Order

Test Scenarios:

Rest API endpoints for different operations:

S. NO	Operation	Endpoints	Method	Access
1	Get Role Details	/api/roles	GET	ADMIN
2	Add a new Role	/api/add/role	POST	ADMIN
3	Get User Details	/api/users	GET	ADMIN
4	Add a new User	/api/add/user	POST	ADMIN
5	Save a new Employee	/api/employee	POST	ADMIN
6	Read Employee List	/api/employees	GET	USER, ADMIN
7	Fetch Employee Data BY Id	/api/employees/{employeeId}	GET	USER, ADMIN
8	Update an existing Employee	/api/employee	PUT	ADMIN
9	Delete an existing Employee	/api/employee/{employeeId}	DELETE	ADMIN
10	Search Employee BY First Name	/api/employees/search/{employeeFirstName}	GET	USER, ADMIN
11	Sort Employee BY First Name IN ASC / DESC Order	/api/employees/sort?order=(ASC/DESC)	GET	USER, ADMIN

Please use below test credentials for Admin and User roles:

Admin:

Username: Manish

Password: admin@123

Test User:

Username: Shyam

Password: user@123

Test Scenario #1: Get Role Details - </api/roles>

The screenshot shows a Postman interface for a GET request to `http://localhost:8080/api/roles`. The request is configured with Basic Authentication, using the username "Manish" and a masked password. The response is a 200 OK status with a 5 ms response time and 394 B of data. The response body is displayed in JSON format, showing an array of two role objects:

```
1 {
2   "id": 1,
3   "name": "ADMIN"
4 },
5 {
6   "id": 2,
7   "name": "USER"
8 }
9
10 ]
```

Test Scenario #2: Add a new Role - </api/add/role>

The screenshot shows a Postman interface for a POST request to `http://localhost:8080/api/add/role`. The request body is a JSON object: `{ "name": "MANAGER" }`. The response is a 200 OK status with a 43 ms response time and 371 B of data. The response body is displayed in JSON format, showing the created role object:

```
1 {
2   "id": 3,
3   "name": "MANAGER"
4 }
```

Test Scenario #3: Get User Details - /api/users

http://localhost:8080/api/users

GET http://localhost:8080/api/users

Auth: Basic Auth

Username: Manish

Password: admin@123

☒ Show Password

Body (JSON):

```
1 {
2   {
3     "id": 1,
4     "userName": "Manish",
5     "password": "$2a$12$ISGmm.2Jnm4.ip3cN0Sbf.2JvbAzrNaujI4RZhnzmY.6jy7atfGa",
6     "roles": [
7       {
8         "id": 1,
9         "name": "ADMIN"
10      }
11    ]
12  },
13  {
14    "id": 2,
15    "userName": "Shyam",
16    "password": "$2a$12$Em/hSqccq8xe8qMcqW5nE.WwACqtPp60ms85zLCIA.dBxj7XaUJS",
17    "roles": [
18      {
19        "id": 2,
20        "name": "USER"
21      }
22    ]
23  }
24 }
```

200 OK 21 ms 619 B

Test Scenario #4: Add a new User - /api/add/user

http://localhost:8080/api/add/user

POST http://localhost:8080/api/add/user

Body (JSON):

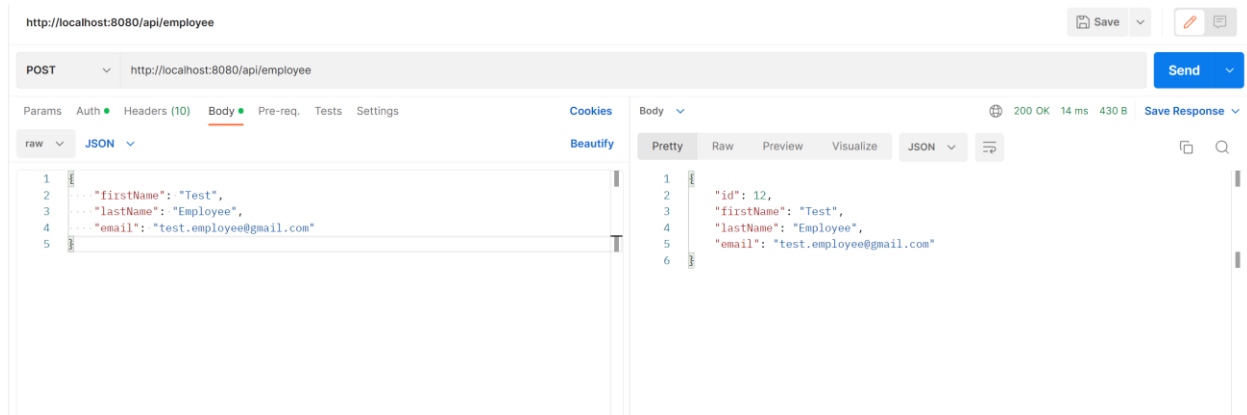
```
1 {
2   "userName": "Temp",
3   "password": "12345",
4   "roles": [
5     {
6       "id": 2,
7       "name": "USER"
8     }
9   ]
10 }
```

Body (JSON):

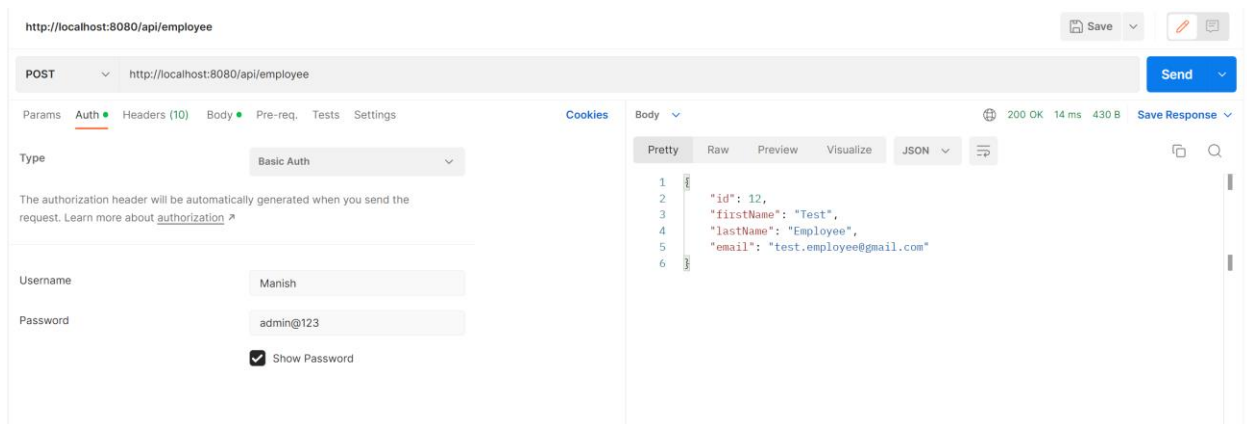
```
1 {
2   "id": 3,
3   "userName": "Temp",
4   "password": "$2a$10$s1E1u0apM1YU5dJemPcGF0e2D6uzPBWnGiJ0145y0mKHYWkyLA2bi",
5   "roles": [
6     {
7       "id": 2,
8       "name": "USER"
9     }
10  ]
11 }
```

200 OK 109 ms 479 B

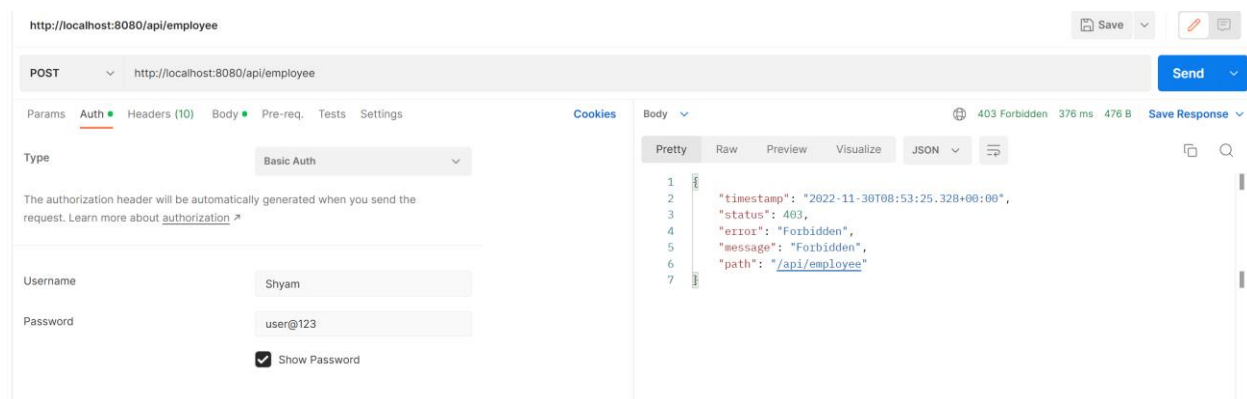
Test Scenario #5: Save a new Employee - </api/employee>



Added above new employee – “Test Employee” using Admin Credentials



Try adding new employee – “Test1234 Employee” using User Credentials, we receive Status 403 “Forbidden”



Test Scenario #6: Read Employee List - `/api/employees`

http://localhost:8080/api/employees

GET http://localhost:8080/api/employees

Params Auth Headers (8) Body Pre-req. Tests Settings

Type Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

Username Manish

Password admin@123

☒ Show Password

Body

```
1 {
2   {
3     "id": 1,
4     "firstName": "Manish",
5     "lastName": "Shivani",
6     "email": "manish.shivani@gmail.com"
7   },
8   {
9     "id": 2,
10    "firstName": "Shyam",
11    "lastName": "Gupta",
12    "email": "shyam.gupta@gmail.com"
13  },
14  {
15    "id": 3,
16    "firstName": "Aman",
17    "lastName": "Nigam",
18    "email": "aman.nigam@gmail.com"
19  },
20  {
21    "id": 4,
22    "firstName": "Devesh",
23    "lastName": "Srivastava",
24    "email": "devesh.srivastava@gmail.com"
25  },
26  {
27    "id": 5,
28    "firstName": "Varun",
29    "lastName": "Sharma",
30    "email": "varun.sharma@gmail.com"
31  },
32  }
```

200 OK 9 ms 1.31 KB Save Response

http://localhost:8080/api/employees

GET http://localhost:8080/api/employees

Params Auth Headers (8) Body Pre-req. Tests Settings

Type Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

Username Shyam

Password user@123

☒ Show Password

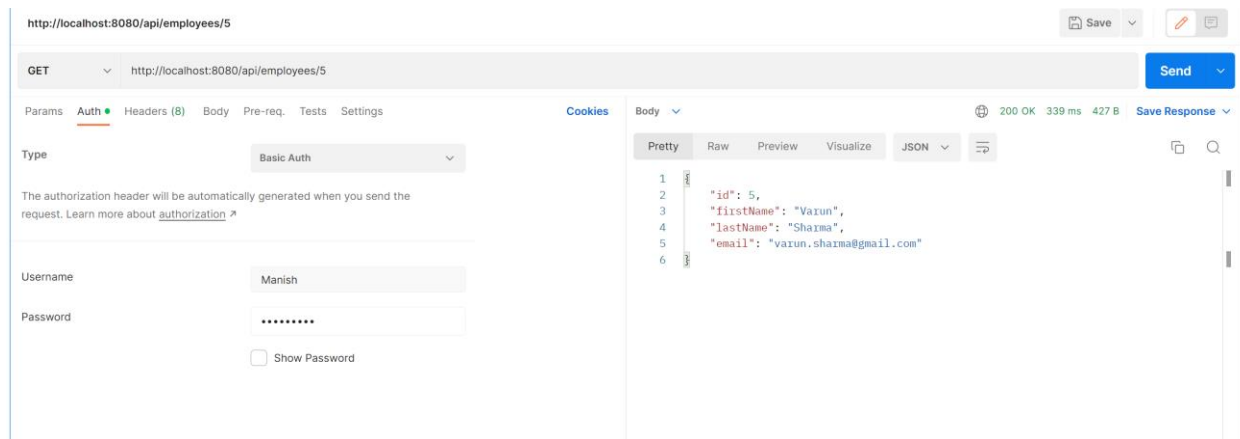
Body

```
1 {
2   {
3     "id": 1,
4     "firstName": "Manish",
5     "lastName": "Shivani",
6     "email": "manish.shivani@gmail.com"
7   },
8   {
9     "id": 2,
10    "firstName": "Shyam",
11    "lastName": "Gupta",
12    "email": "shyam.gupta@gmail.com"
13  },
14  {
15    "id": 3,
16    "firstName": "Aman",
17    "lastName": "Nigam",
18    "email": "aman.nigam@gmail.com"
19  },
20  {
21    "id": 4,
22    "firstName": "Devesh",
23    "lastName": "Srivastava",
24    "email": "devesh.srivastava@gmail.com"
25  },
26  {
27    "id": 5,
28    "firstName": "Varun",
29    "lastName": "Sharma",
30    "email": "varun.sharma@gmail.com"
31  },
32  }
```

200 OK 392 ms 1.31 KB Save Response

Test Scenario #7: Fetch Employee Data BY Id - </api/employees/{employeeId}>

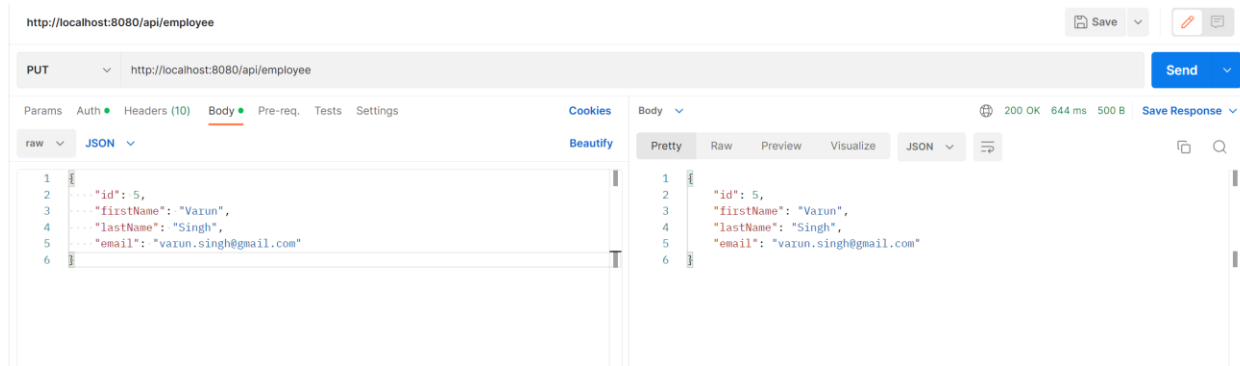
Fetch Employee ID 5:



Test Scenario #8: Update an existing Employee - </api/employee>

For the employee id #5, fetched in Test Scenario 7:

- Update last name from “Sharma” to “Singh”
- Update email from `"varun.sharma@gmail.com"` to `"varun.singh@gmail.com"`



Test Scenario #9: Delete an existing Employee - `/api/employee/{employeeId}`

The screenshot shows the Postman interface for a DELETE request to `http://localhost:8080/api/employee/9`. The request is configured with Basic Auth, Username: Manish, and Password: admin@123. The response is a 200 OK status with a body containing the text "Employee Deleted - ID: 9".

URL: `http://localhost:8080/api/employee/9`

Method: DELETE

Auth: Basic Auth

Username: Manish

Password: admin@123

Response: 200 OK 23 ms 370 B

Body: Employee Deleted - ID: 9

Test Scenario #10: Search Emp BY First Name - `/api/employees/search/{employeeFirstName}`

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/api/employees/search/Mandeep`. The request is configured with Basic Auth, Username: Manish, and Password: The response is a 200 OK status with a body containing a JSON array of employee details.

URL: `http://localhost:8080/api/employees/search/Mandeep`

Method: GET

Auth: Basic Auth

Username: Manish

Password:

Response: 200 OK 10 ms 513 B

Body:

```
[{"id": 5, "firstName": "Mandeep", "lastName": "Singh", "email": "test.singh@gmail.com"}, {"id": 10, "firstName": "Mandeep", "lastName": "Singh", "email": "mandeep.singh@gmail.com"}]
```

Test Scenario #11: Sort Employee BY First Name IN ASC / DESC Order

/api/employees/sort?order=(ASC/DESC)

ASC:

http://localhost:8080/api/employees/sort?order=ASC

GET http://localhost:8080/api/employees/sort?order=ASC

Params Auth Headers (10) Body Pre-req. Tests Settings

Type Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

Username Manish

Password *****

☐ Show Password

Body

```
1 {
2   "id": 8,
3   "firstName": "Alok",
4   "lastName": "Singh",
5   "email": "alok.singh@gmail.com"
6 },
7 {
8   "id": 3,
9   "firstName": "Aman",
10  "lastName": "Nigam",
11  "email": "aman.nigam@gmail.com"
12 },
13 {
14  "id": 7,
15  "firstName": "Ashish",
16  "lastName": "Kohli",
17  "email": "ashish.kohli@gmail.com"
18 },
19 {
20  "id": 4,
21  "firstName": "Devesh",
22  "lastName": "Srivastava",
23  "email": "devesh.srivastava@gmail.com"
24 },
25 {
26  "id": 5,
27  "firstName": "Mandeep",
28  "lastName": "Singh",
29  "email": "test.singh@gmail.com"
30 },
31 }
```

DESC:

http://localhost:8080/api/employees/sort?order=DESC

GET http://localhost:8080/api/employees/sort?order=DESC

Params Auth Headers (10) Body Pre-req. Tests Settings

Type Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

Username Manish

Password *****

☐ Show Password

Body

```
1 {
2   "id": 6,
3   "firstName": "Vikas",
4   "lastName": "Gupta",
5   "email": "vikas.gupta@gmail.com"
6 },
7 {
8   "id": 2,
9   "firstName": "Shyam",
10  "lastName": "Gupta",
11  "email": "shyam.gupta@gmail.com"
12 },
13 {
14  "id": 11,
15  "firstName": "Ramesh",
16  "lastName": "Jha",
17  "email": "ramesh.jha@gmail.com"
18 },
19 {
20  "id": 1,
21  "firstName": "Manish",
22  "lastName": "Shivani",
23  "email": "manish.shivani@gmail.com"
24 },
25 {
26  "id": 5,
27  "firstName": "Mandeep",
28  "lastName": "Singh",
29  "email": "test.singh@gmail.com"
30 },
31 }
```