

1. Put Employee

The screenshot displays a REST client interface with two tabs: **PUT PutEmployee** and **GET GetEmployees**. A third tab, **GET GetEmployeeByID**, is also visible. The **PUT PutEmployee** tab is active, showing the following details:

- Method:** PUT
- URL:** http://localhost:8084/EmpMgt/addEmp
- Body:** The request body is a JSON object with the following fields:

```
{
  "username": "empl",
  "password": "pass1",
  "fullName": "Employee One",
  "emailID": "08/21/1991",
  "dateOfBirth": "empl@email.com",
  "gender": "Male",
  "securityQuestion": "First pet?",
  "securityAnswer": "Dog"
}
```

Below the request details, the **Body** tab is selected, showing the response body in JSON format:

```
{
  "statusCode": 200,
  "status": "Success",
  "message": "Employee data inserted successfully.",
  "data": {
    "employeeId": "27ab664e",
    "username": "empl",
    "password": "pass1",
    "fullName": "Employee One",
    "emailID": "08/21/1991",
    "dateOfBirth": "empl@email.com",
    "gender": "Male",
    "securityQuestion": "First pet?",
    "securityAnswer": "Dog"
  }
}
```

2. Get Employees

The screenshot shows a REST client interface with two tabs: 'PUT PutEmployee' and 'GET GetEmployees'. The 'GET GetEmployees' tab is active, showing a request to 'http://localhost:8084/EmpMgt/getAllEmpDetails'. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests'. The 'Params' tab is selected, showing a table with one parameter: 'Key'. Below this, there are tabs for 'Body', 'Cookies', 'Headers (3)', and 'Test Results'. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The JSON response is as follows:

```
1 {
2   "statusCode": 200,
3   "status": "Success",
4   "message": "",
5   "data": [
6     {
7       "employeeId": "27ab664e",
8       "username": "emp1",
9       "password": "pass1",
10      "fullName": "Employee One",
11      "emailID": "08/21/1991",
12      "dateOfBirth": "emp1@email.com",
13      "gender": "Male",
14      "securityQuestion": "First pet?",
15      "securityAnswer": "Dog"
16    }
17  ]
18 }
```

3. Get Employee By ID – used PostMan Global Variable

The screenshot shows the Postman interface for a REST client. At the top, there are tabs for different requests: 'PUT PutEmployee', 'GET GetEmployees', and 'GET GetEmployeeByID'. The 'GET GetEmployeeByID' tab is selected. Below the tabs, the request method is 'GET' and the URL is 'http://localhost:8084/EmpMgt/getByEmpId/{{idFromCreate}}'. A red box highlights the global variable placeholder '{{idFromCreate}}'. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers (1)', 'Body', 'Pre-request Script', and 'Tests'. The 'Headers (1)' tab is selected, showing a table with one header: 'Accept' with the value 'application/json'. Below the headers, there are tabs for 'Body', 'Cookies', 'Headers (3)', and 'Test Results'. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The JSON response is as follows:

```
1 {
2   "statusCode": 200,
3   "status": "Success",
4   "message": "",
5   "data": {
6     "employeeId": "27ab664e",
7     "username": "emp1",
8     "password": "pass1",
9     "fullName": "Employee One",
10    "emailID": "08/21/1991",
11    "dateOfBirth": "emp1@email.com",
12    "gender": "Male",
13    "securityQuestion": "First pet?",
14    "securityAnswer": "Dog"
15  }
16 }
```

4. Check Login

The screenshot displays a REST client interface with two tabs: 'PUT PutEmployee' and 'POST CheckLogin'. The 'POST CheckLogin' tab is active, showing the URL 'http://localhost:8084/EmpMgt/checkLogin'. The 'Body' tab is selected, showing a JSON payload:

```
{  "username": "emp1",  "password": "pass1"}
```

. Below the request, the 'Test Results' section shows the response body in 'Pretty' format:

```
{  "statusCode": 200,  "status": "Success",  "message": "Employee has authenticated successfully",  "data": []}
```

PUT PutEmployee POST CheckLogin GET GetEmployees

► CheckLogin

POST http://localhost:8084/EmpMgt/checkLogin

Params Authorization Headers (1) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "username": "emp1",
3   "password": "pass1"
4 }
```

Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON

```
1 {
2   "statusCode": 200,
3   "status": "Success",
4   "message": "Employee has authenticated successfully",
5   "data": []
6 }
```

5. Delete Employee

The screenshot shows a REST client interface with two tabs: 'PUT PutEmployee' and 'PUT DeleteEmployeePut'. The 'DeleteEmployeePut' tab is active. Below the tabs, the method 'PUT' is selected, and the URL is 'http://localhost:8084/EmpMgt/deleteEmp/{{idFromCreate}}'. The 'Body' tab is selected, and the 'none' radio button is chosen. The response body is displayed in the 'Body' tab, showing a JSON object with the following fields: 'statusCode' (200), 'status' ('Success'), 'message' ('Employee data deleted successfully.'), and 'data' (an empty array). The response is formatted as JSON and is shown in a 'Pretty' view.

PUT DeleteEmployeePut

PUT http://localhost:8084/EmpMgt/deleteEmp/{{idFromCreate}}

Params Authorization Headers **Body** Pre-request Script Tests

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary

Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON

```
1 {
2   "statusCode": 200,
3   "status": "Success",
4   "message": "Employee data deleted successfully.",
5   "data": []
6 }
```

6. Get Employees after delete

The screenshot shows a REST client interface with the following details:

- Request Method:** GET
- Request URL:** http://localhost:8084/EmpMgt/getAllEmpDetails
- Response Body (JSON):**

```
{
  "statusCode": 200,
  "status": "Success",
  "message": "No Employees data exist",
  "data": []
}
```