**Capstone Project: API Automation**

**https://dummy.restapiexample.com**

**Test Scenarios**

**4.1 Employee Management Endpoints**

**1. GET /api/v1/employees**

**Scenario 1: Retrieve All Employees**

* **Endpoint**: GET /api/v1/employees
* **Steps**:
  1. Send a GET request to retrieve all employees.
  2. Verify the status code is 200.
  3. Validate the response contains a non-empty list of employees.
  4. Check the structure of each employee record (id, employee\_name, employee\_salary, etc.).

**2. GET /api/v1/employee/{id}**

**Scenario 2: Retrieve Employee by ID**

* **Endpoint**: GET /api/v1/employee/{id}
* **Steps**:
  1. Send a GET request to retrieve an employee by a valid ID.
  2. Verify the status code is 200.
  3. Validate the employee's details match the expected values.
  4. Send a GET request with an invalid ID.
  5. Verify the status code is 404 and the response contains an appropriate error message.
* **Scenario 5**: Attempt to retrieve an employee with an invalid ID (e.g., negative number or a large number).
  1. **Endpoint**: GET /api/v1/employee/{id}
  2. **Example**: GET /api/v1/employee/9999
  3. **Expected Result**: Status 404; response contains an error message indicating the employee was not found.
* **Scenario 6**: Validate the response for an employee with an ID that includes special characters.
  1. **Endpoint**: GET /api/v1/employee/{id}
  2. **Example**: GET /api/v1/employee/@!#$
  3. **Expected Result**: Status 400; response contains an error message indicating invalid ID format.

**3. POST /api/v1/create**

**Scenario 7: Create a New Employee**

* **Endpoint**: POST /api/v1/create
* **Steps**:
  1. Send a POST request with valid employee data.
  2. Verify the status code is 200 or 201.
  3. Validate the response contains the newly created employee's details.
  4. Send a POST request with missing or invalid data.
  5. Verify the status code is 400 and the response contains an error message.

**Body**: json

Copy code

{

"name": "John Doe",

"salary": "50000",

"age": "30"

}

* **Scenario 8**: Attempt to create a new employee with missing mandatory fields (e.g., name or salary).
  + **Endpoint**: POST /api/v1/create
  + **Body**:

json

Copy code

{

"name": "",

"salary": "50000",

"age": "30"

}

* + **Expected Result**: Status 400; response contains an error message indicating the missing fields.
* **Scenario 9**: Create a new employee with a large salary value.
  + **Endpoint**: POST /api/v1/create
  + **Body**:

json

Copy code

{

"name": "Rich Person",

"salary": "1000000000",

"age": "45"

}

* + **Expected Result**: Status 201; response contains the details of the newly created employee with the large salary value.

**4. PUT /api/v1/update/{id}**

**Scenario 10: Update an Existing Employee**

* **Endpoint**: PUT /api/v1/update/{id}
* **Steps**:
  1. Send a PUT request to update an employee's details.
  2. Verify the status code is 200.
  3. Validate the response reflects the updated employee details.
  4. Send a PUT request for a non-existent employee ID.
  5. Verify the status code is 404.
* **Scenario 11**: Attempt to update an employee that does not exist.
  1. **Endpoint**: PUT /api/v1/update/{id}
  2. **Example**: PUT /api/v1/update/9999
  3. **Body**:

json

Copy code

{

"name": "Ghost Employee",

"salary": "0",

"age": "0"

}

* 1. **Expected Result**: Status 404; response contains an error message indicating the employee was not found.
* **Scenario 12**: Update an employee's details with an empty name.
  1. **Endpoint**: PUT /api/v1/update/{id}
  2. **Example**: PUT /api/v1/update/1
  3. **Body**:

json

Copy code

{

"name": "",

"salary": "55000",

"age": "31"

}

* 1. **Expected Result**: Status 400; response contains an error message indicating the invalid name field.

**5. DELETE /api/v1/delete/{id}**

**Scenario 13: Delete an Employee**

* **Endpoint**: DELETE /api/v1/delete/{id}
* **Steps**:
  1. Send a DELETE request for an existing employee.
  2. Verify the status code is 200.
  3. Send a GET request to confirm the employee has been deleted.
  4. Send a DELETE request for a non-existent employee ID.
  5. Verify the status code is 404.

}

**GET /api/v1/employees?limit={n}**

* **Scenario 16**: Retrieve a limited number of employee records.
  + **Endpoint**: GET /api/v1/employees?limit=5
  + **Expected Result**: Status 200; response contains a list of 5 employees.
* **Scenario 17**: Retrieve employees with a limit exceeding the total number of employees.
  + **Endpoint**: GET /api/v1/employees?limit=1000
  + **Expected Result**: Status 200; response contains all available employee records.
* **Scenario 18**: Retrieve employees with a limit of 0.
  + **Endpoint**: GET /api/v1/employees?limit=0
  + **Expected Result**: Status 200; response contains an empty list of employees.

**7. GET /api/v1/employees/sort?order={order}**

* **Scenario 19**: Retrieve employees sorted by salary in ascending order.
  + **Endpoint**: GET /api/v1/employees/sort?order=asc
  + **Expected Result**: Status 200; employees are sorted by salary from lowest to highest.
* **Scenario 20**: Retrieve employees sorted by salary in descending order.
  + **Endpoint**: GET /api/v1/employees/sort?order=desc
  + **Expected Result**: Status 200; employees are sorted by salary from highest to lowest.
* **Scenario 21**: Attempt to sort employees with an invalid order value (e.g., order=random).
  + **Endpoint**: GET /api/v1/employees/sort?order=random
  + **Expected Result**: Status 400; response contains an error message indicating the invalid sort order.