API Test Scenario

Restful API Endpoints

```
- GET /login                                       For a valid manager
- PUT /login                                       For a valid manager
- POST /login                                      For a valid manager
- DELETE /login                                    For a valid manager

- GET /users                                       For all users
- PUT /users                                       For all users
- POST /users                                      For all users
- DELETE /users                                    For all users

- GET /users?name={username}                       For a user by username
- PUT /users?name={username}                       For a user by username
- POST /users?name={username}                      For a user by username
- DELETE /users?name={username}                    For a user by username

- GET /users/{id}                                  For a user by ID
- PUT /users/{id}                                  For a user by ID
- POST /users/{id}                                 For a user by ID
- DELETE /users/{id}                               For a user by ID

- GET /users/{id}/configurations                   For all configurations for user
- POST /users/{id}/configurations                  For all configurations for user
- PUT /users/{id}/configurations                   For all configurations for user
- DELETE /users/{id}/configurations                For all configurations for user
```

Scenarios

```
Given /login and /user end points are prepared
When a manager logs in with its valid credentials
When the manager looks through all users list
When the manager creates a new user
When the manager creates the user’s configurations
When the manager selects and see the newly created user's configurations
When the manager removes the created user
When the manager finds the removed user's configurations
When the removed user is not found
```

For the described test scenarios, the following API endpoints will be utilized

When a manager logs in with its valid credentials

Endpoint Used: POST /login

Purpose: This endpoint is used for authenticating a manager with valid credentials. A successful login returns an authorization token that is used for further authenticated API calls

Method: POST

Request Parameters:

- **Headers**: Content-Type: application/json

```
Content-Type: application/json
```

- **Request Body**:

```
{
  "username": "manager_username",
  "password": "manager_password"
}
```

Response:

- **Success**: HTTP 200 with a JSON payload containing an authentication token

```
{
  "token": "auth_token_value"
}
```

- **Failure**:
    - HTTP 401 Indicates invalid credentials

```
{
  "error": "Invalid username or password"
}
```

Test Scenario Implementation:

- Send request to **POST /login** endpoint with valid credentials.
- Capture the token from the response for use in subsequent API calls.
- Assert that the response status is **200** and that the response contains a valid token.

Steps in Gherkin:

```
Feature: Manager Login API Flow

  Scenario Outline: Validate login for a manager
    Given /login endpoint is prepared
    When a manager logs in with its "<username>" and "<password>"
    Then the response status should be 200
    And a token should be returned
```

Explanation: Validates that the manager can successfully log in and receive an authentication token

When the manager looks through all users list

Endpoint Used: GET /users

Purpose: This endpoint is used to retrieve a list of all users. It allows the manager to review user data

Method: GET

Request Parameters:

- **Headers**: Content-Type: application/json, Authorization: Bearer auth_token

```
Content-Type: application/json
Authorization: Bearer auth_token
```

Response:

- **Success**: HTTP 200 with a JSON array containing user details

```
[
  { "id": "user_id_1", "name": "User 1", ... },
  { "id": "user_id_2", "name": "User 2", ... }
]
```

- **Failure**:
  - HTTP 401 Indicates that the request is unauthorized if the token is missing or invalid

Test Sceanrio Implementation:

- Send a request to **GET /users** endpoint with the valid Authorization header
- Assert that the response status is **200** and verify that the response body contains the expected user list

Steps in Gherkin:

```
Given /login and /user end points are prepared
When the manager looks through all users list
Then the response status should be 200 and a list of users should be returned
```

Explanation: Validates that the manager can access the list of users

When the manager creates a new user

Endpoint Used: POST /users

Purpose: This endpoint is used to create a new user. It allows the manager to add new user

Method: POST

Request Parameters:

- **Headers**: Content-Type: application/json, Authorization: Bearer auth_token

```
Content-Type: application/json
Authorization: Bearer auth_token
```

- **Request Body**:

```
{
  "name": "John Doe",
  "email": "JohnDoe@test.com",
  "role": "employee"
}
```

Response:

- **Success**: HTTP 201 with the details of the newly created user

```
{
  "id": 1,
  "name": "John Doe",
  "email": "JohnDoe@test.com",
  "role": "employee"
}
```

- **Failure**:
  - HTTP 401 Indicates that the request is unauthorized if the token is missing or invalid
  - HTTP 400 Bad Request (Indicates invalid input)

Test Sceanrio Implementation:

- Send a request to **POST /users** endpoint with the valid Authorization header and a JSON payload containing the user details
- Assert that the response status is **201** and verify that the response body contains the created user data

Steps in Gherkin:

```
Given /login and /user end points are prepared
When the manager creates a new user
Then the response status should be 201 and the user details should be returned
```

Explanation: Create and validate that the manager can create a new user successfully

When the manager creates the user’s configuration

Endpoint Used: POST /users/{id}/configurations

Purpose: The manager uses it to create configurations for the newly created user

Method: POST

Request Parameters:

- **Headers**: Content-Type: application/json, Authorization: Bearer auth_token

```
Content-Type: application/json
Authorization: Bearer auth_token
```

- **Path Parameter**: The unique identifier of the user whose configurations need to be created

```
/users/1/configurations
```

- **Request Body:**

```
{
  "configKey": "example_key",
  "configValue": "example_value"
}
```

Response:

- **Success**: HTTP 201 with the details of the created configuration

```
{
  "id": 1,
  "configKey": "example_key",
  "configValue": "example_value"
}
```

- **Failure**:
  - HTTP 401 Indicates that the request is unauthorized if the token is missing or invalid
  - HTTP 400 Bad Request (Indicates invalid input)

Test Sceanrio Implementation:

- Send a request to **POST /users/{id}/configurations** endpoint with the valid Authorization header and a JSON payload containing the configuration details
- Assert that the response status is **201** and verify that the response body contains the created configuration details

Steps in Gherkin:

```
Given /login and /user end points are prepared
When the manager creates the user's configurations
Then the response status should be 201 and the configuration details should be returned
```

Explanation: Validates that the manager can create configurations for a specified user

When the manager selects and see the newly created user's configurations

Precondition: Get the list of all users using **GET /users** and retrieve the **ID** of the newly created user

Endpoint Used: GET /users/{id}/configurations

Purpose: Allows the manager to retrieve user's configurations

Method: GET

Request Parameters:

- **Headers**: Content-Type: application/json, Authorization: Bearer auth_token

```
Content-Type: application/json
Authorization: Bearer auth_token
```

- **Path Parameter**: The unique identifier of the user whose configurations need to be created

```
/users/1/configurations
```

Response:

- **Success**: HTTP 200 with a JSON array of configurations

```
{
  "id": 1,
  "configKey": "example_key",
  "configValue": "example_value"
}
```

- **Failure**:
  - HTTP 401 Indicates that the request is unauthorized if the token is missing or invalid
  - HTTP 404 Not Found (Indicates invalid ID)

Test Sceanrio Implementation:

- Send a request to **GET /users/{id}/configurations** endpoint with the valid Authorization header
- Assert that the response status is **200** and verify that the response body contains the configurations for the specified user

Steps in Gherkin:

```
Given /login and /user end points are prepared
When the manager selects and sees the newly created user's configurations
Then the response status should be 200 and the configurations should be displayed
```

Explanation: Validates the manager can view the configurations of a specified user

When the manager removes the created user

Endpoint Used: DELETE /users/{id}

Purpose: This endpoint is used to remove a specific user by ID

Method: DELETE

Request Parameters:

- **Headers**: Content-Type: application/json, Authorization: Bearer auth_token

```
Content-Type: application/json
Authorization: Bearer auth_token
```

- **Path Parameter**: The unique identifier of the user whose configurations need to be created.

```
/users/1
```

Response:

- **Success**: HTTP 200 or 204 (No Content), indicating successful deletion
- **Failure:**
  - HTTP 404 Not Found (Indicates invalid ID)

- HTTP 401 Indicates that the request is unauthorized if the token is missing or invalid

Test Sceanrio Implementation:

- Send a request to **DELETE /users/{id}** endpoint with the valid Authorization header
- Assert that the response status is **200** or **204** and verify that the user is no longer present

Steps in Gherkin:

```
Given /login and /user end points are prepared
When the manager removes the created user
Then the response status should be 200 or 204
And the user should not be found when accessed afterward
```

Explanation: Validates that the manager can delete a user successfully

When the manager finds the removed user's configurations

When the manager attempts to find the configurations of a removed user, we need to check that the system properly handles this situation by returning an appropriate response indicating that the configurations cannot be found

Endpoint Used: GET /users/{id}/configurations

Purpose: Allows the manager to retrieve user's configurations

Method: GET

Request Parameters:

- **Headers**: Content-Type: application/json, Authorization: Bearer auth_token

```
Content-Type: application/json
Authorization: Bearer auth_token
```

- **Path Parameter**: The unique identifier of the user whose configurations need to be created

```
/users/1/configurations
```

Response:

- **Failure:**
  - HTTP 404 Indicates that the configurations for the user are not found, confirming that the user was deleted and any related data is no longer accessible

```
{
  "error": "Configurations not found for the user"
}
```

  - HTTP 401 Indicates that the request is unauthorized if the token is missing or invalid

Test Sceanrio Implementation:

- Delete the user using **DELETE /users/{id}**, capture the **id** used for the deletion
- Use the **id** to call **GET /users/{id}/configurations** to get user configurations
- Assert that the response status is **404** and that the response body contains an appropriate error message

Steps in Gherkin:

```
Given /login and /user end points are prepared
When the manager finds the removed user's configurations
Then the response status should be 404 and the error message should indicate "Configurations not found for the user"
```

Explanation: The verification step ensures that when a user is deleted, their configurations are no longer accessible

When the removed user is not found

We need to validate that the API returns the correct response indicating that the user no longer exists after deletion

Endpoint Used: GET /users/{id}

Purpose: Allows the manager to retrieve user's configurations

Method: GET

Request Parameters:

- **Headers**: Content-Type: application/json, Authorization: Bearer auth_token

```
Content-Type: application/json
Authorization: Bearer auth_token
```

- **Path Parameter**: The unique identifier of the user whose configurations need to be created

```
/users/1
```

Response:

- **Success:** HTTP 200 returns the user information
- **Failure:**
  - HTTP 404 Indicates that the user is not found, confirming the deletion

```
{
  "error": "User not found"
}
```

  - HTTP 401 Indicates that the request is unauthorized if the token is missing or invalid

Test Sceanrio Implementation:

- Send a request to **GET /users/{id}** endpoint with the valid Authorization header after the user has been deleted.
- Assert that the response status is **404** and the response body or error message reflects that the user does not exist

Steps in Gherkin:

```
Given /login and /user end points are prepared
When the manager checks for the removed user
Then the response status should be 404
And the response should indicate that the user is not found
```

Explanation: Ensures that after a user is deleted, attempts to access the user's details results in a **404** Not Found status