Insider API Task

▼ Test Automation - API

- Using “pet” endpoints from https://petstore.swagger.io/ write CRUD operations API tests with
positive and negative scenarios.

▼ User Story

- As a user, I should be able to perform CRUD operations (Create, Read, Update, Delete) on pets, so that I can manage the pets in the system effectively

  Environment / Base URL : https://petstore.swagger.io/

  Component : PET- Everything about your Pets

▼ Acceptance Criteria

1. **Add a new pet to the store (Create)**
   - **Positive**: Verify that the user can create a new pet using a POST request with valid data.
   - **Negative**: Verify that the user cannot create a new pet with invalid or missing required fields in the POST request.
2. **Find pet by ID (Read)**
   - **Positive**: Verify that the user can retrieve a pet's information using a GET request with a valid pet ID.
   - **Negative**: Verify that the user receives an appropriate error when trying to retrieve a pet with an invalid or non-existent pet ID using a GET request.
3. **Update an existing pet (Update)**
   - **Positive**: Verify that the user can update pet details using a PUT request with valid data.
   - **Negative**: Verify that the user cannot update pet details with invalid data or missing required fields in the PUT request.

4. **Delete a pet (Delete)**
    - **Positive**: Verify that the user can delete a pet using a DELETE request with a valid pet ID.
    - **Negative**: Verify that the user receives an error when trying to delete a pet with an invalid or non-existent pet ID using a DELETE request.

▼ TEST CASES

▼ AC 1: Add a new pet to the store (Create)

Positive Test Case:

- **Test Case ID**: TC_Create_01
- **Test Objective**: Create a new pet with valid data.
- body * required
- **Test Steps**:
    1. Send a POST request to `/pets` with the following valid pet data:

```
{
  "id": 21,
  "category": {
    "id": 0,
    "name": "Kangal"
  },
  "name": "Kara",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

    2. Check the response code is 200 (Created).

3. Verify the response body contains the pet's information with a unique pet id

```
{
    "id": 21,
    "category": {
        "id": 0,
        "name": "Kangal"
    },
    "name": "Kara",
    "photoUrls": [
        "string"
    ],
    "tags": [
        {
            "id": 0,
            "name": "string"
        }
    ],
    "status": "available"
}
```

- **Expected Result**: A new pet is successfully created and the response contains the correct pet details.

Negative Test Case:

- **Test Case ID**: TC_Create_02
- **Test Objective**: Attempt to create a pet with missing or invalid data.
- **Test Steps**:
  1. Send a GET request to `/pets` with valid data

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "Kangal"
  },
```

```
  "name": "Ak",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

2. Check the response code is 405 (Method NOT Allowed).
3. Verify the response body contains an error message

```
{
    "code": 405,
    "type": "unknown"
}
```

- **Expected Result**: The request is rejected with a 405 error and a clear error message.

▼ AC 2: Find pet by ID (Read)

Positive Test Case:

- **Test Case ID**: TC_Read_01
- **Test Objective**: Retrieve a pet's information using a valid pet ID.
- **Pre-Conditions**: A pet exists in the store.
- **Test Steps**:
  1. Send a GET request to `/pets/21` with a valid pet ID.
  2. Check the response code is 200 (OK).
  3. Verify the response body contains the correct pet information,

```
{
  "id": 21,
  "category": {
    "id": 0,
    "name": "Kangal"
  },
  "name": "Kara",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```


- **Expected Result**: The pet information is successfully retrieved, and the details are correct.

Negative Test Case:

- **Test Case ID**: TC_Read_02
- **Test Objective**: Attempt to retrieve a pet with an invalid or non-existent pet ID.
- **Test Steps**:
  1. Send a GET request to `/pets/-21` with an invalid or non-existent pet ID.
  2. Check the response code is 404 (Not Found).
  3. Verify the response body contains an error message indicating the pet was not found


```
{
  "code": 1,
  "type": "error",
```

```
  "message": "Pet not found"
}
```

- **Expected Result**: The request is rejected with a 404 error and an appropriate error message.

AC 3: Update an existing pet (Update)

Positive Test Case:

- **Test Case ID**: TC_Update_01
- **Test Objective**: Update pet details with valid data.
- **Test Steps**:
  1. Send a PUT request to `/pets/21` with valid data to update the pet

```
{
  "id": 21,
  "category": {
    "id": 0,
    "name": "Çomar"
  },
  "name": "Ak",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

  2. Check the response code is 200 (OK).
  3. Verify the response body contains the updated pet information,

```
{
  "id": 21,
```

```
  "category": {
    "id": 0,
    "name": "Çomar"
  },
  "name": "Ak",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

- **Expected Result**: The pet's details are successfully updated, and the new information is reflected.

Negative Test Case:

- **Test Case ID**: TC_Update_02
- **Test Objective**: Attempt to update pet details with invalid data.
- **Test Steps**:
  1. Send a PUT request to `/pets` with invalid or incomplete data

     ```
     {}
     ```

  2. Check the response code is 400 (Bad Request).
  3. Verify the response body contains an error message indicating invalid data

     ```
     {
         "id": 9222968140497181625,
         "photoUrls": [],
     ```

```
    "tags": []
}
```

- **Expected Result**: The request is rejected with a 200 OK → TEST FAILED

▼ AC 4: Delete a pet (Delete)

Positive Test Case:

- **Test Case ID**: TC_Delete_01
- **Test Objective**: Delete a pet using a valid pet ID.
- **Pre-Conditions**: A pet exists in the store.
- **Test Steps**:
  1. Send a DELETE request to `/pets/21` with a valid pet ID
  2. Add api_key → string - header (NOT Required)
  3. Check the response code is 200 .
  4. Send a GET request to `/pets/21` to verify that the pet no longer exists.

```
{
    "code": 200,
    "type": "unknown",
    "message": "21"
}
```

- **Expected Result**: The pet is successfully deleted 200 OK,
- And a GET request for the pet ID returns a 404 Not Found error.

```
{
    "code": 1,
    "type": "error",
    "message": "Pet not found"
}
```

Negative Test Case:

- **Test Case ID**: TC_Delete_02
- **Test Objective**: Attempt to delete a pet with an invalid or non-existent pet ID.
- **Test Steps**:
    1. Send a DELETE request to `/pets/21` with an invalid or non-existent pet ID.
    2. Check the response code is 404 (Not Found).
    3. Verify the response body contains an error message indicating the pet was not found.**Example Response**:

- **Expected Result**: The request is rejected with a 404 error and an appropriate error message.