INSIDER API TASK TEST CASES

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

USER STORY

As a QA Engineer, I should be able to test the CRUD operations for the “pet” component in the Petstore API, so that I can ensure the endpoints are functioning correctly and handle both positive and negative scenarios.

BaseURL= **https://petstore.swagger.io/v2**

ACCEPTANCE CRITERIAS

AC01: Add a new pet to the petStore

> **AC01.a:** A new pet can be added to the store by POST request to the “/pet” endpoint with valid json body.
>
> **AC01.b:** Attempt to add a pet with invalid or missing required fields returns appropriate errors.

AC02: Update an existing pet

> **AC02.a:** An existing pet can be updated by PUT request to the “/pet” endpoint with valid json body.
>
> **AC02.b:** Attempt to update a pet with invalid or nonexistent id returns appropriate errors.

AC03: Find Pets by status

> **AC03.a:** Existing pets can be retrieved based on their status (“available”, “pending”, and “sold” status options) by GET request to the “/pet/findByStatus” endpoint with the selected status as query parameter.
>
> **AC03.b:** Invalid status or missing parameters return appropriate errors.

AC04: Find pet by ID

> **AC04.a:** An existing pet can be retrieved by GET request to the “/pet/{petId}” endpoint with its ID as path parameter.
>
> **AC04.b:** Attempt to retrieve a pet with an invalid or nonexistent id returns appropriate errors.

AC05: Update a pet in the store with form data

> **AC05.a:** An existing pet can be updated by POST request to the “/pet/{petId}” endpoint with its ID as path parameter, name and status as formData.
>
> **AC05.b:** Attempt to update a pet with invalid or missing form data returns appropriate errors.

AC06: Upload an image for an existing pet

> **AC06.a:** An image can be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.
>
> **AC06.b:** Attempt to upload a file with an invalid format or size returns appropriate errors.

AC07: Delete existing pet

> **AC07.a:** An existing pet can be deleted by DELETE request to the “/pet/{petId}” endpoint with its ID as path parameter and api_key token (not required in Swagger) in request header.
>
> **AC07.b:** Attempt to delete a pet with an invalid or nonexistent id returns appropriate errors.

TEST CASES

TEST CASES FOR AC01: POST Add a new pet to the petStore

POSITIVE TEST CASE FOR AC01:

> **TC01:** Verify a new pet can be added to the store by POST request to the “/pet” endpoint with valid json body.

1- Send a POST request to the “/pet” endpoint with valid json body which contains;

```
{
  "id": 45140,
  "category": {
                 "id": 0,
                "name": "Max"
            },
  "name": "Maxym",
  "photoUrls": [
                "http://foo.bar.com/1"
             ],
  "tags": [
           {
             "id": 0,
              "name": "kitty"
           }
        ],
  "status": "available"
}
```

2- Verify that the response status code is 200 and response body contains data from request body.

NEGATIVE TEST CASES FOR AC01:

TC02: Verify a new pet can not be added to the store by POST request to the “/pet” endpoint with empty response body.

1- Send a POST request to the “/pet” endpoint with empty json body.
2- Verify that the response status code is 415 Unsupported Media Type.

TC03: Verify a new pet can not be added to the store by POST request to the “/pet” endpoint with invalid json body format.

1- Send a POST request to the “/pet” endpoint with invalid json body format which contains;

```
{
  "id": 45140,
  "category": ,
  "name": ,
  "photoUrls": [
                    "http://foo.bar.com/1"
                ],
  "tags": [
             {
               "id": 0,
                "name": "kitty"
             }
          ],
  "status": "available"
}
```

2- Verify that the response status code is 400 Bad Request.

TC04: Verify a new pet can not be added to the store by POST request to the “/pet” endpoint with invalid json body format.

1- Send a POST request to the “/pet” endpoint with id as String in json body format which contains;

```
{
  "id": “45140”,
  "category": {
                     "id": 0,
                    "name": "Max"
                },
  "name": "Maxym",
  "photoUrls": [
                    "http://foo.bar.com/1"
                ],
  "tags": [
          {
            "id": 0,
            "name": "kitty"
          }
        ],
  "status": "available"
}
```

2- Verify that the response status code is 400 Bad Request.

TC05: Verify a new pet can not be added to the store by GET request to the "/pet" endpoint with valid json body format.

1- Send a GET request to the "/pet" endpoint with valid json body format which contains;

```
{
  "id": 45140,
  "category": {
                "id": 0,
               "name": "Max"
              },
  "name": "Maxym",
  "photoUrls": [
               "http://foo.bar.com/1"
              ],
  "tags": [
          {
            "id": 0,
            "name": "kitty"
          }
        ],
  "status": "available"
}
```

2- Verify that the response status code is 405 Method Not Allowed.

TC06: Verify a new pet can not be added to the store by PATCH request to the "/pet" endpoint with valid json body format.

1- Send a PATCH request to the "/pet" endpoint with valid json body format.
2- Verify that the response status code is 405 Method Not Allowed.

TC07: Verify a new pet can not be added to the store by DELETE request to the "/pet" endpoint with valid json body format.

1- Send a DELETE request to the "/pet" endpoint with valid json body format.
2- Verify that the response status code is 405 Method Not Allowed.

TEST CASES FOR AC02: PUT Update an existing pet

POSITIVE TEST CASE FOR AC02:

TC08: Verify an existing pet can be updated by PUT request to the “/pet” endpoint with valid json body.

1- Send a POST request to the “/pet” endpoint with valid json body which contains;

```
{
  "id": 45140,
  "category": {
                         "id": 0,
                        "name": "MaxThunder"
                 },
  "name": "Maxym",
  "photoUrls": [
                        "http://foo.bar.com/2"
                   ],
  "tags": [
              {
                "id": 0,
                 "name": "kittyBoy"
              }
          ],
  "status": "sold"
}
```

2- Verify that the response status code is 200 and response body contains data from request body.

NEGATIVE TEST CASES FOR AC02:

TC09: Verify an existing pet can NOT be updated by PUT request to the “/pet” endpoint with empty response body.

1- Send a PUT request to the “/pet” endpoint with empty json body.
2- Verify that the response status code is 415 Unsupported Media Type.

TC10: Verify an existing pet can NOT be updated by PUT request to the “/pet” endpoint with invalid json body format.

1- Send a PUT request to the “/pet” endpoint with invalid json body format which contains;

```
{
  "id": 45140,
  "category": ,
  "name": ,
  "photoUrls": [
                  "http://foo.bar.com/1"
               ],
  "tags": [
            {
              "id": 0,
               "name": "kitty"
            }
          ],
  "status": "available"
}
```

2- Verify that the response status code is 400 Bad Request.

TC11: Verify an existing pet can NOT be updated by PUT request to the “/pet” endpoint with invalid json body format.

1- Send a PUT request to the “/pet” endpoint with id as String in json body format which contains;

```
{
  "id": “45140”,
  "category": {
                  "id": 0,
                 "name": "Max"
              },
  "name": "Maxym",
  "photoUrls": [
                  "http://foo.bar.com/1"
               ],
  "tags": [
            {
              "id": 0,
               "name": "kitty"
            }
          ],
  "status": "available"
}
```

2- Verify that the response status code is 400 Bad Request.

TC12: Verify an existing pet can NOT be updated by GET request to the “/pet” endpoint with invalid json body format.

1- Send a GET request to the “/pet” endpoint with valid json body format which contains;

```
{
  "id": 45140,
  "category": {
```

```
                         "id": 0,
                        "name": "Max"
                  },
    "name": "Maxym",
    "photoUrls": [
                        "http://foo.bar.com/1"
                   ],
    "tags": [
               {
                 "id": 0,
                  "name": "kitty"
               }
           ],
    "status": "available"
  }
```

2- Verify that the response status code is 405 Method Not Allowed.

TC13: Verify an existing pet can NOT be updated by PATCH request to the “/pet” endpoint with valid json body format.

1- Send a PATCH request to the “/pet” endpoint with valid json body format.
2- Verify that the response status code is 405 Method Not Allowed.

TC14: Verify an existing pet can NOT be updated by DELETE request to the “/pet” endpoint with valid json body format.

1- Send a DELETE request to the “/pet” endpoint with valid json body format.
2- Verify that the response status code is 405 Method Not Allowed.

TC15: Verify an existing pet can NOT be updated by PUT request to the “/pet” endpoint with invalid pet ID.

1- Send a PUT request to the “/pet” endpoint with invalid pet ID.
2- Verify that the response status code is 404 Pet Not Found.

TEST CASES FOR AC03: **GET** Find Pets by status

POSITIVE TEST CASES FOR AC03:

TC16: Verify existing pets can be retrieved based on their “available” status by GET request to the “/pet/findByStatus” endpoint with “available” as query parameter.

1- Send a GET request to the “/pet/findByStatus” endpoint with “available” as query param.

2. Verify that the response status code is 200 and response body contains existing pets with “available” status.

TC17: Verify existing pets can be retrieved based on their “pending” status by GET request to the “/pet/findByStatus” endpoint with “pending” as query parameter.

1. Send a GET request to the “/pet/findByStatus” endpoint with “pending” as query param.
2. Verify that the response status code is 200 and response body contains existing pets with “pending” status.

TC18: Verify existing pets can be retrieved based on their “sold” status by GET request to the “/pet/findByStatus” endpoint with “sold” as query parameter.

1. Send a GET request to the “/pet/findByStatus” endpoint with “sold” as query param.
2. Verify that the response status code is 200 and response body contains existing pets with “sold” status.

NEGATIVE TEST CASES FOR AC03:

TC19: Verify existing pets can NOT be retrieved based on their status by GET request to the “/pet/findByStatus” endpoint with invalid status query parameter.

1. Send a GET request to the “/pet/findByStatus” endpoint with “zold” as query param.
2. Verify that the response status code is 400 Invalid Status.

TC20: Verify existing pets can NOT be retrieved based on their status by GET request to the “/pet/findByStatus” endpoint with missing part or character as query parameter.

1. Send a GET request to the “/pet/findByStatus” endpoint with “availabl” as query param.
2. Verify that the response status code is 400 Invalid Status.

TC21: Verify existing pets can NOT be retrieved based on their status by GET request to the “/pet/findByStatus” endpoint with empty status as query parameter.

1. Send a GET request to the “/pet/findByStatus” endpoint with empty status “” as query param.
2. Verify that the response status code is 400 Invalid Status.

TEST CASES FOR AC04: GET Find pet by ID

POSITIVE TEST CASE FOR AC04:

TC22: Verify an existing pet can be retrieved by GET request to the “/pet/{petId}” endpoint with valid pet ID as path parameter.

1- Send a GET request to the “/pet/{petId}” endpoint with “45140” as path param.
2- Verify that the response status code is 200 and response body contains pet details.

NEGATIVE TEST CASES FOR AC04:

TC23: Verify a non-existing pet can NOT be retrieved by GET request to the “/pet/{petId}” endpoint with non-existing pet ID as path parameter.

1- Send a GET request to the “/pet/{petId}” endpoint with invalid pet ID “45141” as path param.
2- Verify that the response status code is 404 Not Found and response body contains “Pet not found” message.

TC24: Verify an existing pet can NOT be retrieved by GET request to the “/pet/{petId}” endpoint with empty pet ID as path parameter.

1- Send a GET request to the “/pet/{petId}” endpoint with empty pet ID “” as path param.
2- Verify that the response status code is 405 Method Not Allowed.

TC25: Verify an existing pet can NOT be retrieved by GET request to the “/pet/{petId}” endpoint with string chars in pet ID as path parameter.

1- Send a GET request to the “/pet/{petId}” endpoint with string chars in pet ID “one” as path param.
2- Verify that the response status code is 404 Not Found and response body contains “java.lang.NumberFormatException: For input string: \"one\"” message.

TC26: Verify an existing pet can NOT be retrieved by GET request to the “/pet/{petId}” endpoint with special chars in pet ID as path parameter.

1- Send a GET request to the “/pet/{petId}” endpoint with special chars in pet ID “” as path param.
2- Verify that the response status code is 404 Not Found and response body contains “java.lang.NumberFormatException: For input string: \"=!!$$()\"” message.

TC27: Verify an existing pet can NOT be retrieved by GET request to the “/pet/{petId}” endpoint with negative integer chars in pet ID as path parameter.

1- Send a GET request to the “/pet/{petId}” endpoint with negative integer chars in pet ID “-45140” as path param.
2- Verify that the response status code is 404 Not Found and response body contains “Pet not found” message.

TEST CASES FOR AC05: POST Update a pet in the store with form data

POSITIVE TEST CASES FOR AC05:

TC28: Verify an existing pet can be updated by POST request to the “/pet/{petId}” endpoint with its ID as path parameter, name and status “available” as formData.

1- Send a POST request to the “/pet/{petId}” endpoint with valid pet ID “45140” as path param, string name “Bony” as fromData and string status “available” as fromData.
2- Verify that the response status code is 200 and response body contains pet ID “45140” as message.

TC29: Verify an existing pet can be updated by POST request to the “/pet/{petId}” endpoint with its ID as path parameter, name and status “pending” as formData.

1- Send a POST request to the “/pet/{petId}” endpoint with valid pet ID “45140” as path param, string name “Bony” as fromData and string status “pending” as fromData.
2- Verify that the response status code is 200 and response body contains pet ID “45140” as message.

TC30: Verify an existing pet can be updated by POST request to the “/pet/{petId}” endpoint with its ID as path parameter, name and status “sold” as formData.

1- Send a POST request to the “/pet/{petId}” endpoint with valid pet ID “45140” as path param, string name “Bony” as fromData and string status “sold” as fromData.
2- Verify that the response status code is 200 and response body contains pet ID “45140” as message.

NEGATIVE TEST CASES FOR AC05:

TC31: Verify an existing pet can NOT be updated by POST request to the “/pet/{petId}” endpoint with its NEGATIVE INTEGER pet ID as path parameter, name and status “available” as formData.

1- Send a POST request to the “/pet/{petId}” endpoint with its NEGATIVE INTEGER pet ID “-45140” as path param, string name “Bony” as fromData and string status “available” as fromData.

2- Verify that the response status code is 404 Not Found and response body contains “Pet not found” as message.

TC32: Verify a non-existing pet can NOT be updated by POST request to the “/pet/{petId}” endpoint with non-existing pet ID as path parameter, name and status “available” as formData.

1- Send a POST request to the “/pet/{petId}” endpoint with non-existing pet ID “45141” as path param, string name “Bony” as fromData and string status “available” as fromData.
2- Verify that the response status code is 404 Not Found and response body contains “not found” as message.

TC33: Verify an existing pet can NOT be updated by POST request to the “/pet/{petId}” endpoint with Special Chars pet ID as path parameter, name and status “available” as formData.

1- Send a POST request to the “/pet/{petId}” endpoint with Special Chars pet ID “-_?=)” as path param, string name “Bony” as fromData and string status “available” as fromData.
2- Verify that the response status code is 404 Not Found and response body contains “java.lang.NumberFormatException: For input string: \"-_\"” as message.

TC34: Verify an existing pet can NOT be updated by POST request to the “/pet/{petId}” endpoint with valid pet ID as path parameter, name and status details in json body format.

1- Send a POST request to the “/pet/{petId}” endpoint with valid pet ID “45140” as path param, string name “Bony” and string status “available” as inside json body as;
```
{
   "name":"Bouncy",
   "status":"pending"
}
```
2- Verify that the response status code is 415 Unsupported Media Type.

TC35: Verify an existing pet can NOT be updated by POST request to the “/pet/{petId}” endpoint with valid pet ID as path parameter, name and status details in text body format.

1- Send a POST request to the “/pet/{petId}” endpoint with valid pet ID “45140” as path param, string name “Bony” and string status “available” as inside text body as;
```
{
   "name":"Bouncy",
   "status":"pending"
}
```

2- Verify that the response status code is 415 Unsupported Media Type.

TC36: Verify an existing pet can NOT be updated by POST request to the “/pet/{petId}” endpoint with valid pet ID as path parameter, name and status details in text body format.

1- Send a POST request to the “/pet/{petId}” endpoint with valid pet ID “45140” as path param, string name “Bony” and string status “available” as inside text body as;
```
{
   "name":"Bouncy",
   "status":"pending"
}
```
2- Verify that the response status code is 415 Unsupported Media Type.

TC37: Verify an existing pet can NOT be updated by PUT request to the “/pet/{petId}” endpoint with valid pet ID as path parameter, name and status “available” as formData.

1- Send a PUT request to the “/pet/{petId}” endpoint with valid pet ID “45140” as path param, string name “Maxy” as fromData and string status “available” as fromData.
2- Verify that the response status code is 405 Method Not Allowed.

TC38: Verify an existing pet can NOT be updated by PATCH request to the “/pet/{petId}” endpoint with valid pet ID as path parameter, name and status “available” as formData.

1- Send a PATCH request to the “/pet/{petId}” endpoint with valid pet ID “45140” as path param, string name “Maxy” as fromData and string status “available” as fromData.
2- Verify that the response status code is 405 Method Not Allowed.

TEST CASES FOR AC06: **POST** Upload an image for an existing pet

POSITIVE TEST CASES FOR AC06:

TC39: Verify a valid image file in .png format can be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, string additionalMetadata “A png file” and file “API.png” as formData.

2- Verify that the response status code is 200 and response body contains message “additionalMetadata: A png file\nFile uploaded to ./API.png, 582245 bytes”.

TC40: Verify a valid image file in .jpg format can be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, string additionalMetadata “A jpg file” and file “API.jpg” as formData.
2- Verify that the response status code is 200 and response body contains message “additionalMetadata: A jpg file\nFile uploaded to ./API.jpg, 582245 bytes”.

TC41: Verify a valid image file in .png format can be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, and file “API.png” as formData.
2- Verify that the response status code is 200 and response body contains message “additionalMetadata: null\nFile uploaded to ./API.png, 582245 bytes”.

TC42: Verify a valid image file in .jpg format can be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, and file “API.jpg” as formData.
2- Verify that the response status code is 200 and response body contains message “additionalMetadata: null\nFile uploaded to ./API.jpg, 582245 bytes”.

NEGATIVE TEST CASES FOR AC06:

TC43: Verify an excessively large .png file can NOT be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, string additionalMetadata “A png file” and an excessively large .png file “API.png” as formData.
2- Verify that the response status code is 413 Request Entity Too Large.

TC44: Verify an excessively large .jpg file can NOT be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, string additionalMetadata “A jpg file” and an excessively large .jpg file “API.jpg” as formData.
2- Verify that the response status code is 413 Request Entity Too Large.

TC45: Verify a .txt file can NOT be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, string additionalMetadata “A txt file” and a .txt file “API.txt” as formData.
2- Verify that the response status code is 415 Unsupported Media Type.

TC46: Verify a .mp3 file can NOT be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, string additionalMetadata “An mp3 file” and a .txt file “API.mp3” as formData.
2- Verify that the response status code is 415 Unsupported Media Type.

TC47: Verify a .mp4 file can NOT be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, string additionalMetadata “An mp4 file” and a .txt file “API.mp4” as formData.
2- Verify that the response status code is 415 Unsupported Media Type.

TC48: Verify a .docx file can NOT be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, string additionalMetadata “A docx file” and a .docx file “API.docx” as formData.
2- Verify that the response status code is 415 Unsupported Media Type.

TC49: Verify a .xslx file can NOT be uploaded for an existing pet by POST request to the “/pet/{petId}/uploadImage” endpoint with its ID as path parameter, additionalMetadata and file as formData.

1- Send a POST request to the “/pet/{petId}/uploadImage” endpoint with valid pet ID “45140” as path param, string additionalMetadata “A xslx file” and a .xslx file “API.xslx” as formData.
2- Verify that the response status code is 415 Unsupported Media Type.

TEST CASES FOR AC07: **DELETE** Delete existing pet

POSITIVE TEST CASES FOR AC07:

TC50: Verify an existing pet can be deleted by DELETE request to the “/pet/{petId}” endpoint with its ID as path parameter and api_key token in request header.

1- Send a DELETE request to the “/pet/{petId}” endpoint with valid pet ID “45140” as path param, and api_key token in request header.
2- Verify that the response status code is 200.

NEGATIVE TEST CASES FOR AC07:

TC51: Verify an existing pet can NOT be deleted by DELETE request to the “/pet/{petId}” endpoint with invalid ID format (String) as path parameter and api_key token in request header.

1- Send a DELETE request to the “/pet/{petId}” endpoint with invalid pet ID in String format “four” as path param, and api_key token in request header.
2- Verify that the response status code is 404 Not Found.

TC52: Verify an existing pet can NOT be deleted by DELETE request to the “/pet/{petId}” endpoint with invalid ID format (Special Chars) as path parameter and api_key token in request header.

1- Send a DELETE request to the “/pet/{petId}” endpoint with invalid pet ID with Special Chars such as “?!-=)” as path param, and api_key token in request header.
2- Verify that the response status code is 404 Not Found.

TC53: Verify an existing pet can NOT be deleted by DELETE request to the “/pet” endpoint with query parameter and api_key token in request header.

1- Send a DELETE request to the “/pet” endpoint with query param such as “/pet?petId=45140”, and api_key token in request header.

2- Verify that the response status code is 405 Method Not Allowed.

TC54: Verify a non-existing pet can NOT be deleted by DELETE request to the “/pet/{petId}” endpoint with path parameter and api_key token in request header.

1- Send a DELETE request to the “/pet/{petId}” endpoint with a non-existing pet ID as path param, and api_key token in request header.
2- Verify that the response status code is 404 Not Found.