Description of Test Cases Run

This document contains descriptions of test cases that have been run on the API endpoints in the format that is given as part of the problem statement, and also screenshots of a few unit tests run using PyTest.

Adhil Ahmed P M Shums	**Afnan Ahmad**	**Preetodeep Dev**
21F1003972	21F1003730	21F1005636

---

1. Endpoint: GET */api/v1/tags?tagIDList=all*

Input: The query parameter *"tagIDList*" specifies the tag IDs that we need to query. If "*all*" is specified, then every tag would be returned.

Expected Output: All the tags that are there in the database. For example:

```
{   "status": 200,
    "message": "Request successful",
    "tags": [
        {   "tagID": 1,
            "name": "t12022",
            "description": "Posts created on term 1
(January-April) of 2022"
        },
        {   "tagID": 2,
            "name": "portal",
            "description": "Posts related to the seek portal"
        },
        {   "tagID": 3,
            "name": "oppe",
            "description": "OPPE-related posts"
        }
   ]
}
```

Actual Output: All the tags that have been created in the test database.


```
{   "status": 200,
    "message": "Request successful",
    "tags": [
        {   "tagID": 1,
            "name": "t12022",
            "description": "Posts created on term 1
(January-April) of 2022"
        },
        {   "tagID": 2,
            "name": "portal",
            "description": "Posts related to the seek portal"
        },
        {   "tagID": 3,
            "name": "oppe",
            "description": "OPPE-related posts"
        }
   ]
}
```


Result: SUCCESS

2. Endpoint: GET */api/v1/tags?tagIDList=9,10,11*

Input: “9,10,11” input is passed as query parameter “tagIDList” in the URL

Expected Output: Tags that are there in the database with the given IDs 9,10, and 11.


```
{   "status": 200,
    "message": "Request successful",
    "tags": [
        {   "tagID": 9,
            "name": "quiz1",
            "description": "Posts related to quiz 1"
        },
        {   "tagID": 10,
            "name": "quiz2",
            "description": "Posts related to quiz 2"
        },
        {   "tagID": 11,
            "name": "end-term",
            "description": "Posts related to the end-term
exam"
        }
   ]
}
```


Actual Output: Tags that are there in the database with the given IDs 9,10, and 11.


```
{   "status": 200,
    "message": "Request successful",
    "tags": [
        {   "tagID": 9,
            "name": "quiz1",
            "description": "Posts related to quiz 1"
        },
        {   "tagID": 10,
```

```
            "name": "quiz2",
            "description": "Posts related to quiz 2"
        },
        {   "tagID": 11,
            "name": "end-term",
            "description": "Posts related to the end-term
exam"
        }
    ]
}
```


Result: SUCCESS

3. Endpoint: GET */api/v1/tags?tagIDList=35*

Input: “35” input is passed as query parameter “tagIDList” in the URL. This is designed to test for a non-existent tag being requested from the API.

Expected Output: The API should return a 404 error, saying “no matching tags found”.

```
{
    "status": 404,
    "message": "No matching tags found!"
}
```

Actual Output: The API returns a 404 error, saying “no matching tags found”.

```
{
    "status": 404,
    "message": "No matching tags found!"
}
```

Result: SUCCESS

4. Endpoint: POST */api/v1/tags*

Input: The following input is passed request body:

```
{
    "name": “TestTagCreation”,
    "description": "Test check for tag creation"
}
```

Expected Output: The API should return a 201 code along with the newly created tag as a JSON object.

```
{
    "status": 201,
    "tagID": 14,
    "name": “TestTagCreation”,
    "description": "Test check for tag creation"

}
```

Actual Output: The API returns a 201 code along with the newly created tag as a JSON object.

```
{
    "status": 201,
    "tagID": 14,
    "name": “TestTagCreation”,
    "description": "Test check for tag creation"

}
```

Result: SUCCESS

5. Endpoint: POST */api/v1/tickets*

Input: The following input is passed to the request body. A valid authorization token is also passed in the request header.

```
{
  "title": "Test Ticket Title",
  "firstMessage": "This is the first message of test ticket",
  "tags": "8,7,4",
  "visibility": "public"
}
```

Expected Output: The API should return a 201 code along with the newly created ticket as a JSON object.

```
{
  "status": 201,
  "message": "Ticket created successfully",
  "ticketID": "1",
  "title": "Sample Title",
  "messageID": "1",
  "firstMessage": "This is the first message",
  "visibility": "public",
  "senderID": "2",
  "senderName": "UnitTester",
  "timestamp": "09/04/2023 22:30"
}
```

Actual Output: The API returns a 201 code along with the newly created ticket as a JSON object.

```
{
  "status": 201,
  "message": "Ticket created successfully",
  "ticketID": "1",
```

```
    "title": "Sample Title",
    "messageID": "1",
    "firstMessage": "This is the first message",
    "visibility": "public",
    "senderID": "2",
    "senderName": "UnitTester",
    "timestamp": "09/04/2023 22:30"
}
```


Result: SUCCESS

6. Endpoint: POST */api/v1/tickets*

Input: The following malformed input is passed to the request body. A valid authorization token is also passed in the request header.

```
{
  "title": "Test Ticket Title"
}
```

Expected Output: The API should return a 400 code along with an error message.

```
{
  "status": 400,
  "message": "Malformed request",
}
```

Actual Output: The API returns a 400 code along with an error message.

```
{
  "status": 400,
  "message": "Malformed request",
}
```

Result: SUCCESS

7. Endpoint: POST */api/v1/tickets*

Input: The following input is passed to the request body. No authorization token is passed in the request header.

```
{
  "title": "Test Ticket Title",
  "firstMessage": "This is the first message of test ticket",
  "tags": "8,7,4",
  "visibility": "public"
}
```

Expected Output: The API should return a 403 code along with an error message wrapped in a JSON object

```
{
  "status": 403,
  "message": "No access rights, forbidden!"
}
```

Actual Output: The API returns a 403 code along with the error message wrapped in a JSON object

```
{
  "status": 403,
  "message": "No access rights, forbidden!"
}
```

Result: SUCCESS

8. Endpoint: GET */api/v1/tickets*

Input: The authorization token is passed in the request header.

Expected Output: The API should return a 200 code along with all the public tickets wrapped in a JSON object

```
{
  "status": 200,
  "message": "Request successful",
  "tickets": [
    {
      "ticketID": "1",
      "Votes": 6,
      "title": "Ticket Title 1",
      "Status": "In progress",
      "LastResponseTime": "<time_in_format>"
    },
    {
      "ticketID": "2",
      "Votes": 4,
      "title": "Ticket Title 2",
      "Status": "Open",
      "LastResponseTime": "<time_in_format>"
    },
    {
      "ticketID": "3",
      "Votes": 8 ,
      "title": "Ticket Title 3",
      "Status": "Closed",
      "LastResponseTime": "<time_in_format>"
    }
  ]
}
```

Actual Output: The API returns a 200 code along with all the public tickets wrapped in a JSON object


```
{
  "status": 200,
  "message": "Request successful",
  "tickets": [
    {
      "ticketID": "4",
      "Votes": 6,
      "title": "Ticket Title 4",
      "Status": "In progress",
      "LastResponseTime": "<time_in_format>"
    },
    {
      "ticketID": "5",
      "Votes": 2,
      "title": "Ticket Title 5",
      "Status": "Open",
      "LastResponseTime": "<time_in_format>"
    },
    {
      "ticketID": "6",
      "Votes": 7,
      "title": "Ticket Title 6",
      "Status": "Closed",
      "LastResponseTime": "<time_in_format>"
    }
  ]
}
```


Result: SUCCESS

9. Endpoint: GET */api/v1/mytickets*

Input: The authorization token is passed in the request header.

Expected Output: The API should return a 200 code along with all the tickets created by the logged-in user (as identified via the supplied authorization token) wrapped in a JSON object


```
{
  "status": 200,
  "message": "Request successful",
  "tickets": [
     {
     "ticketID": "4",
     "Votes": 6,
     "title": "TestUser Ticket 1",
     "Status": "In progress",
     "Visibility": "Public",
     "LastResponseTime": "09/04/2023 22:35"
     },
     {
     "ticketID": "7",
     "Votes": 1,
     "title": "TestUser Ticket 2",
     "Status": "In progress",
     "Visibility": "Private",
     "LastResponseTime": "09/04/2023 21:13"
     },
     {
     "ticketID": "8",
     "Votes": 2,
     "title": "TestUser Ticket 3",
     "Status": "In progress",
     "Visibility": "Public",
     "LastResponseTime": "09/04/2023 20:01"
     }
  ]
}
```

Actual Output: The API returns a 200 code along with all the tickets created by the logged-in user (as identified via the supplied authorization token) wrapped in a JSON object


```
{
  "status": 200,
  "message": "Request successful",
  "tickets": [
     {
     "ticketID": "4",
     "Votes": 6,
     "title": "TestUser Ticket 1",
     "Status": "In progress",
     "Visibility": "Public",
     "LastResponseTime": "09/04/2023 22:35"
     },
     {
     "ticketID": "7",
     "Votes": 1,
     "title": "TestUser Ticket 2",
     "Status": "In progress",
     "Visibility": "Private",
     "LastResponseTime": "09/04/2023 21:13"
     },
     {
     "ticketID": "8",
     "Votes": 2,
     "title": "TestUser Ticket 3",
     "Status": "In progress",
     "Visibility": "Public",
     "LastResponseTime": "09/04/2023 20:01"
     }
  ]
}
```


Result: SUCCESS

10. Endpoint: PUT */api/v1/tickets/<int:ticket_id:>*

Input: The following input is passed to the request body. A valid authorization token is also passed in the request header.

```
{
  "title": "Sample Title",
  "visibility": "public"
}
```

Expected Output: The API should return a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Ticket modified successfully"
}
```

Actual Output: The API returns a 200 code along with the success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Ticket modified successfully"
}
```

Result: SUCCESS

11. Endpoint: DELETE */api/v1/tickets/<int:ticket_id:*

Input: A valid authorization token is passed in the request header.

Expected Output: The API should return a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Ticket deleted successfully"
}
```

Actual Output: The API returns a 200 code along with the success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Ticket deleted successfully"
}
```

Result: SUCCESS

12. Endpoint: GET */api/v1/messages?start=1?count=3?ticket_id=4*

Input: The query parameter *"start*" specifies the index of the fist message to be loaded, the query parameter *"count*" specifies the number of messages to be loaded and the query parameter *"ticket_id*" specifies which ticket's messages we need to query.

Expected Output: The API should return a 200 code along with the requested messages


```
{
  "status": 200,
  "message": "Request successful",
  "ticketID": "4",
  "LastResponseTime": "09/04/2023 13:01"
,
  "messages": [
    {
      "messageID": "1",
      "text": "Message Text 1",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 12:59"
    },
    {
      "messageID": "2",
      "text": "Message Text 2",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 13:00"
    },
    {
      "messageID": "3",
      "text": "Message Text 3",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 13:01"
    }
  ]
```

```
}
```

Actual Output: The API returns a 200 code along with the requested messages wrapped in a JSON object


```
{
  "status": 200,
  "message": "Request successful",
  "ticketID": "4",
  "LastResponseTime": "09/04/2023 13:01"
,
  "messages": [
    {
      "messageID": "1",
      "text": "Message Text 1",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 12:59"
    },
    {
      "messageID": "2",
      "text": "Message Text 2",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 13:00"
    },
    {
      "messageID": "3",
      "text": "Message Text 3",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 13:01"
    }
  ]
}
```


Result: SUCCESS

---

13. Endpoint: DELETE */api/v1/messages/<int:message_id>*

Input: A valid authorization token is passed in the request header. Message ID 4 is passed in the URL for deletion.

Expected Output: The API should return a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Article deleted successfully",
  "articleID": "4"
}
```

Actual Output: The API should returns a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Article deleted successfully",
  "articleID": "4"
}
```

Result: SUCCESS

14. Endpoint: POST */api/v1/articles*

Input: A valid authorization token is passed in the request header. Article contents are passed in the request body.

```
{
  "title": "Test Article Title",
  "content": "Test Article Content",
  "visibility": "public"
}
```

Expected Output: The API should return a 201 code along with a success message and the newly created article object wrapped in a JSON object.

```
{
  "status": 201,
  "content": "Article created successfully",
  "articleID": "2",
  "title": "Test Article Title",
  "content": "Test Article Content",
  "visibility": "public"
}
```

Actual Output: The API returns a 201 code along with a success message and the newly created article object wrapped in a JSON object.

```
{
  "status": 201,
  "content": "Article created successfully",
  "articleID": "2",
  "title": "Test Article Title",
  "content": "Test Article Content",
  "visibility": "public"
}
```

Result: SUCCESS

---

15. Endpoint: DELETE */api/v1/admin/articles/<int:article_id:>*

Input: A valid authorization token is passed in the request header.

Expected Output: The API should return a 200 code along with a success message and the article ID wrapped in a JSON object

```
{
  "status": 200,
  "message": "Article deleted successfully",
  "articleID": "1"
}
```

Actual Output: The API returns a 200 code along with the success message and the article ID wrapped in a JSON object

```
{
  "status": 200,
  "message": "Article deleted successfully",
  "articleID": "1"
}
```

Result: SUCCESS

16. Endpoint: POST */api/v1/admin/comments*

Input: The following input is passed to the request body. A valid authorization token is also passed in the request header.

```
{
  "article_id": "1",
  "content": "Comment content 1",
  "hidden": "False"
}
```

Expected Output: The API should return a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 201,
  "message": "Comment created successfully",
  "articleID": "1",
  "commentID": "1",
  "content": "Comment content 1"
}
```

Actual Output: The API returns a 200 code along with the success message wrapped in a JSON object

```
{
  "status": 201,
  "message": "Comment created successfully",
  "articleID": "1",
  "commentID": "1",
  "content": "Comment content 1"
}
```

Result: SUCCESS

A FEW TEST CASES RUN USING PYTEST - ALL PASSED

```
(venv) afnan@AFNAN-PC:/mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy_4apd/src$ pytest test_api.py
========================== test session starts ==========================
platform linux -- Python 3.10.6, pytest-7.3.0, pluggy-1.0.0
rootdir: /mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy_4apd/src
collected 3 items

test_api.py ...                                                    [100%]

=============================================== 3 passed in 4.36s ===============================================
(venv) afnan@AFNAN-PC:/mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy_4apd/src$
```

```
(venv) afnan@AFNAN-PC:/mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy
_4apd/src$ pytest test_api.py
========================== test session starts ==========================
platform linux -- Python 3.10.6, pytest-7.3.0, pluggy-1.0.0
rootdir: /mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy_4apd/src
collected 3 items

test_api.py ...                                                    [100%]

========================== 3 passed in 4.02s ==========================
(venv) afnan@AFNAN-PC:/mnt/c/Users/Afnan/Documents/GitHub/soft-engg-pro
ject-jan-2023-group-4-synergy_4apd/src$
```