Test Case - Smart Support

Submitted by :Group-13 (21f1003002, 21f1003442, 21f1003481)

We have implemented API testing using **pytest** for the CRUD APIs for the **Ticket** model in our application

```python
def test_login():
    global JWT
    response = app.test_client().post('/api/user/login', json={
        "username": "craig_fox",
        "password": "password"
    })
    JWT = response.get_json()['access_token']
    assert response.status_code == 200


def test_get_tickets():
    response =  app.test_client().get(
        '/api/tickets?page=0&per_page=10', headers={'Authorization': 'Bearer ' + JWT})
    assert response.status_code == 200


def test_post_tickets():
    global TICKET_ID
    response =  app.test_client().post(
        '/api/tickets', headers={'Authorization': 'Bearer ' + JWT},
        json={
            "title": "Need help with Python",
            "body": "I'm having trouble understanding Python classes. Can someone help me?"
        })
    TICKET_ID = response.get_json()['ticket_id']
    print(TICKET_ID)
    assert response.status_code == 200


def test_get_ticket_by_id():
    response =  app.test_client().get(
        '/api/tickets/{}'.format(TICKET_ID), headers={'Authorization': 'Bearer ' + JWT})
    assert response.status_code == 200
```

So far we have implemented 6 test cases for Ticket APIs (including 1 for the Login API)

```
(project) navin@DESKTOP-J1K386M:~/iit/SE/project/soft-engg-project-jan-2023-group-13/project/code/backend/smartSupport$ pytest -v
========================================== test session starts ==========================================
platform linux -- Python 3.11.0rc1, pytest-7.2.2, pluggy-1.0.0 -- /home/navin/iit/SE/project/.venv/bin/python
cachedir: .pytest_cache
rootdir: /home/navin/iit/SE/project/soft-engg-project-jan-2023-group-13/project/code/backend/smartSupport
plugins: Faker-18.4.0
collected 6 items

tests/test_tickets_api.py::test_login PASSED                                                    [ 16%]
tests/test_tickets_api.py::test_get_tickets PASSED                                              [ 33%]
tests/test_tickets_api.py::test_post_tickets PASSED                                             [ 50%]
tests/test_tickets_api.py::test_get_ticket_by_id PASSED                                         [ 66%]
tests/test_tickets_api.py::test_update_ticket_by_id PASSED                                      [ 83%]
tests/test_tickets_api.py::test_delete_ticket_by_id PASSED                                      [100%]

========================================== 6 passed in 1.43s ==========================================
(project) navin@DESKTOP-J1K386M:~/iit/SE/project/soft-engg-project-jan-2023-group-13/project/code/backend/smartSupport$
```

Some other test cases that we are considering

Test to get all comments for a valid ticket ID

- **Page being tested**: get_comments function
- **Inputs**: A valid ticket ID in the URL and an authenticated JWT token
- **Expected output**: A JSON response with status code 200 and a list of all comments for the given ticket ID in descending order of creation time
- **Actual output**: Same as expected output
- **Result**: Success

Test to get a single comment by ID

- **Page being tested**: get_comment function
- **Inputs**: A valid comment ID in the URL and an authenticated JWT token
- **Expected output**: A JSON response with status code 200 and the comment details for the given ID
- **Actual output**: Same as expected output
- **Result**: Success

Test to add a new comment to a valid ticket

- **Page being tested**: post_comment function
- **Inputs**: A valid ticket ID in the URL, a valid JWT token, and a JSON payload with a "body" field containing the comment text
- **Expected output**: A JSON response with status code 200 and the details of the newly created comment, including the comment ID and creation time
- **Actual output**: Same as expected output
- **Result**: Success

Test to update an existing comment

- **Page being tested**: put_comment function
- **Inputs**: A valid comment ID in the URL, a valid JWT token, and a JSON payload with a "body" field containing the updated comment text
- **Expected output**: A JSON response with status code 200 and the details of the updated comment, including the comment ID and creation time
- **Actual output**: Same as expected output
- **Result**: Success

Test to delete an existing comment

- **Page being tested**: delete_comment function
- **Inputs**: A valid comment ID in the URL and a valid JWT token
- **Expected output**: A JSON response with status code 204 and no content
- **Actual output**: Same as expected output
- **Result**: Success

Test to mark a comment as the solution for a ticket

- **Page being tested**: mark_comment_as_solution function
- **Inputs**: A valid comment ID in the URL and a valid JWT token
- **Expected output**: A JSON response with status code 200 and the details of the marked comment, including the comment ID and creation time. The associated ticket status should also be updated to "Resolved"
- **Actual output**: Same as expected output
- **Result**: Success

Besides the tests mentioned in previous pages, there can be tests written for:

- User CRUD operations
- Tickets Voting operations
- Tags CRUD operations
- FAQs CRUD operations