CS3001- Software Engineering (Jan-May 2024)

Milestone-5 Report

Group 21

- Isha Nayar : CS21B035
- Janapati Varshita Devi : CS21B036
- Kumar Kshitiz Singh : CS21B044
- Lagudu Sree Teja Vardhan : CS20B046
- Nisanth D : CS20B057

INDEX

1.1 App Testing

1.1.1 App Setup

Testing helps ensure that the app will work as expected for the end users.

Software projects which are tested properly and following standard practices, is a good indicator of the quality of the software. Unit tests test the functionality of an individual unit of code isolated from its dependencies. They are the first line of defense against errors and inconsistencies in the codebase.

For this project few of the unit tests are considered. These unit tests consist of testing API endpoints for discourse and webhooks integration for the ticketing system. To carry out testing, ‘PyTest’ python library is used.

The unit tests are divided into 3 major components

- Auth component testing
- Thread component testing

1.1.2 Sample Fixture

We are using the same fixture generated by the existing ticketing system code as part of our API testing. This fixture starts the app with test configuration and within app context the tests are carried out.

```python
from application import create_app
import pytest
from application.logger import logger

# --------------------  Constants  --------------------

# Please set following required constants to mimic a specific user role.

# STUDENT
student_user_id = "ccec26f5a52560cd22a2965287dc4ad9"
student_web_token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImJvc

# SUPPORT
support_user_id = "a5997f803b4dfbdb0a7f17b012ca1697"
support_web_token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImJvc

# ADMIN
admin_user_id = "3ad51db3c4defba9c7f1ca7549712e25"
admin_web_token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImFiY2F

# --------------------  Code  --------------------

# before testing set current dir to `\code\backend`
@pytest.fixture(scope='module')
def test_client():
    flask_app = create_app(env_type="test")
    logger.info("Testing fixture set.")

    # Create a test client using the Flask application configured for testi
    with flask_app.test_client() as testing_client:
        # Establish an application context
        with flask_app.app_context():
            yield testing_client  # this is where the testing happens!
```

1.1.3 Sample Test

The test case code contains request URL, inputs, headers with user id and web token.The docstring explains test case importance.

The following figure shows sample test code

```python
def test_thread_generation_post_200_success(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/thread/create_thread/{ticket_id}' page is requested (POST)
    by student and ticket_id isnt mapped already
    THEN check that the response is 200
    """
    headers = {
        "Content-type": "application/json",
        "web_token": student_web_token,
        "user_id": student_user_id,
    }

    response = test_client.post(
        f"/api/{API_VERSION}/thread/create_thread/{ticket_id}",
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 200
```

1.2 Test Description

1.2.1 Authorization Testing

Test: POST Request on Register Page

Description	GIVEN a Flask application configured for testing WHEN the '/api/v1/auth/register' page is requested (POST) with all correctly filled data fields for a new user THEN check that the response is 200 i.e. the account is created successfully
Page Being Tested	'/api/v1/auth/register'
Inputs	f"/api/{API_VERSION}/auth/register", json={ "first_name": "tej", "last_name": "v1", "email": random_email, "password": "1234", "retype_password": "1234",

	"role": "student", "discourse_id": random_id, "discourse_api_key": "1234",
Expected Output	Status code: 200
Actual Output	Status code: 200
Results	Passed

Test: POST Request on Register Page

Description	GIVEN a Flask application configured for testing WHEN the '/api/v1/auth/register' page is requested (POST) with existing discourse id THEN check that the response is 409 i.e. Discourse id already exists
Page Being Tested	'/api/v1/auth/register'
Inputs	f"/api/{API_VERSION}/auth/register", json={ "first_name": "tej", "last_name": "v1", "email": random_email, "password": "1234", "retype_password": "1234", "role": "student", "discourse_id": random_id, "discourse_api_key": "1234",
Expected Output	Status code: 409 # discourse id already exists
Actual Output	Status code: 409
Results	Passed

Test: POST Request on Register Page

Description	GIVEN a Flask application configured for testing WHEN the '/api/v1/auth/register' page is requested (POST) with unmatched API-key with discourse id THEN check that

	the response is 401 i.e. Discourse id does not matches with the API-key provided, incorrect API-key
Page Being Tested	'/api/v1/auth/register'
Inputs	f"/api/{API_VERSION}/auth/register", json={ "first_name": "tej", "last_name": "v1", "email": random_email, "password": "1234", "retype_password": "1234", "role": "student", "discourse_id": random_id, "discourse_api_key": "1234",
Expected Output	401 # discourse API Key invalid
Actual Output	Status code: 401
Results	Passed

Test: POST Request on Thread Page

Description	GIVEN a Flask application configured for testing WHEN the '/api/v1/thread/create_thread/{ticket_id}' page is requested (POST) by student and ticket_id is not mapped already THEN check that the response is 200
Page Being Tested	'/api/v1/thread/create_thread/{ticket_id}'
Inputs	f"/api/{API_VERSION}/thread/create_thread/{ticket_id}",headers=headers, test_client
Expected Output	Status code: 200
Actual Output	Status code: 200
Results	Passed

Test: POST Request on Thread Page

Description	GIVEN a Flask application configured for testing WHEN the '/api/v1/thread/create_thread/{ticket_id}' page is requested (POST) by student and ticket_id already exists THEN check that the response is 403
Page Being Tested	'/api/v1/thread/create_thread/{ticket_id}'
Inputs	f"/api/{API_VERSION}/thread/create_thread /{ticket_id}",headers=headers, test_client
Expected Output	Status code: 403
Actual Output	Status code: 403
Results	Passed

Test: PUT Request on Thread Page

Description	GIVEN a Flask application configured for testing WHEN the '/api/v1/thread/close_thread/{thread_id}' page is requested (PUT) by student and thread_id is valid THEN check that the response is 200
Page Being Tested	'/api/v1/thread/close_thread/{ticket_id}'
Inputs	f"/api/{API_VERSION}/thread/close_thread/{ ticket_id}",headers=headers, test_client
Expected Output	Status code: 200
Actual Output	Status code: 200
Results	Passed

Test: PUT Request on Thread Page

Description	GIVEN a Flask application configured for testing WHEN the '/api/v1/thread/close_thread/{thread_id}' page is requested (PUT) by student and thread_id is invalid THEN check that the response is 404
Page Being Tested	'/api/v1/thread/close_thread/{ticket_id}'
Inputs	f"/api/{API_VERSION}/thread/close_thread/{ticket_id}",headers=headers, test_client
Expected Output	Status code: 404
Actual Output	Status code: 404
Results	Passed

Test: PUT Request on Thread Page

Description	GIVEN a Flask application configured for testing WHEN the '/api/v1/thread/reopen_thread/{thread_id}' page is requested (PUT) by student and thread_id is closed already THEN check that the response is 200
Page Being Tested	'/api/v1/thread/reopen_thread/{ticket_id}'
Inputs	f"/api/{API_VERSION}/thread/reopen_thread/{ticket_id}",headers=headers, test_client
Expected Output	Status code: 200
Actual Output	Status code: 200
Results	Passed

Test: PUT Request on Thread Page

Description	GIVEN a Flask application configured for testing WHEN the '/api/v1/thread/reopen_thread/{thread_id}' page is requested (PUT) by student and thread_id is invalid or not closed THEN check that the response is 404
Page Being Tested	'/api/v1/thread/reopen_thread/{ticket_id}'
Inputs	f"/api/{API_VERSION}/thread/reopen_thread/{ticket_id}",headers=headers, test_client
Expected Output	Status code: 404
Actual Output	Status code: 404
Results	Passed

1.3 Final Test Summary

Screenshots of each test case code is attached below

Sample Screenshot - 1

```python
def test_register_page_with_fixture_post_200_success(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/auth/register' page is requested (POST)
    with all correctly filled data fields for a new user
    THEN check that the response is 200 i.e. the account is created successfully
    """

    random_email = f"tej{str(int(time.time()))}@gmail.com"
    random_id = f"{str(int(time.time()))}tej"

    response = test_client.post(
        f"/api/{API_VERSION}/auth/register",
        json={
            "first_name": "tej",
            "last_name": "v1",
            "email": random_email,
            "password": "1234",
            "retype_password": "1234",
            "role": "student",
            "discourse_id": random_id,
            "discourse_api_key": "1234",
        },
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 200  # account created successfully
```

Sample Screenshot - 2

```python
def test_register_page_with_fixture_post_409_discourse_id_exists(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/auth/register' page is requested (POST)
    with existing discourse id
    THEN check that the response is 409 i.e. Discourse id already exists
    """

    random_email = f"tej{str(int(time.time()))}@gmail.com"

    response = test_client.post(
        f"/api/{API_VERSION}/auth/register",
        json={
            "first_name": "tej",
            "last_name": "v1",
            "email": random_email,
            "password": "1234",
            "retype_password": "1234",
            "role": "student",
            "discourse_id": "1210tej",
            "discourse_api_key": "1234",
        },
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 409  # discourse id already exists
```

Sample Screenshot - 3

```python
def test_register_page_with_fixture_post_401_incorrect_APIKey(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/auth/register' page is requested (POST) with
    unmatched API-key with discourse id
    THEN check that the response is 401 i.e.
    Discourse id doesnot matches with the API-key provided, incorrect API-key
    """

    random_email = f"tej{str(int(time.time()))}@gmail.com"

    response = test_client.post(
        f"/api/{API_VERSION}/auth/register",
        json={
            "first_name": "tej",
            "last_name": "v1",
            "email": random_email,
            "password": "1234",
            "retype_password": "1234",
            "role": "student",
            "discourse_id": "nisanth",
            "discourse_api_key": "0000",
        },
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 401  # discourse API Key invalid
```

Sample Screenshot - 4

```python
def test_thread_generation_post_200_success(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/thread/create_thread/{ticket_id}' page is requested (POST)
    by student and ticket_id isnt mapped already
    THEN check that the response is 200
    """
    headers = {
        "Content-type": "application/json",
        "web_token": student_web_token,
        "user_id": student_user_id,
    }

    response = test_client.post(
        f"/api/{API_VERSION}/thread/create_thread/{ticket_id}",
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 200
```

Sample Screenshot -5

```python
def test_thread_closing_put_404_invalid_thread_id(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/thread/close_thread/{thread_id}' page is
    requested (PUT) by student and thread_id is invalid
    THEN check that the response is 404
    """
    headers = {
        "Content-type": "application/json",
        "web_token": student_web_token,
        "user_id": student_user_id,
    }

    response = test_client.put(
        f"/api/v1/thread/close_thread/{thread_id}",
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 404
```

Sample Screenshot - 6

```
def test_thread_reopening_put_404_invalid_thread_id(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/thread/reopen_thread/{thread_id}' page is
    requested (PUT) by student and thread_id is invalid or not closed
    THEN check that the response is 404
    """
    headers = {
        "Content-type": "application/json",
        "web_token": student_web_token,
        "user_id": student_user_id,
    }

    response = test_client.put(
        f"/api/v1/thread/reopen_thread/{thread_id}",
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 404
```

Sample Screenshot - 7

```python
def test_thread_generation_post_200_success(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/thread/create_thread/{ticket_id}' page is requested (POST)
    by student and ticket_id isnt mapped already
    THEN check that the response is 200
    """
    headers = {
        "Content-type": "application/json",
        "web_token": student_web_token,
        "user_id": student_user_id,
    }

    response = test_client.post(
        f"/api/{API_VERSION}/thread/create_thread/{ticket_id}",
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 200
```

Sample Screenshot - 8

```python
def test_thread_closing_put_404_invalid_thread_id(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/thread/close_thread/{thread_id}' page is
    requested (PUT) by student and thread_id is invalid
    THEN check that the response is 404
    """
    headers = {
        "Content-type": "application/json",
        "web_token": student_web_token,
        "user_id": student_user_id,
    }

    response = test_client.put(
        f"/api/v1/thread/close_thread/{thread_id}",
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 404
```

Sample Screenshot - 9

```python
def test_thread_reopening_put_404_invalid_thread_id(test_client):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/api/v1/thread/reopen_thread/{thread_id}' page is
    requested (PUT) by student and thread_id is invalid or not closed
    THEN check that the response is 404
    """
    headers = {
        "Content-type": "application/json",
        "web_token": student_web_token,
        "user_id": student_user_id,
    }

    response = test_client.put(
        f"/api/v1/thread/reopen_thread/{thread_id}",
        headers=headers,
    )
    response = response.get_json()
    assert response["status"] == 404
```