Test Log - Logging Framework with Python

Test Environment:

- Operating System: Windows 11
- Python Version: 3.10
- Flask Version: 2.2.3

Test Cases:

1. Register Endpoint

Objective: Verify the functionality of the '/register' endpoint to register a new user.

Steps:

1. Send a POST request to the '/register' endpoint with the following payload:

{ "username": "test_user", "password": "password123" }

   - *Expected Result:* The request is successful (HTTP 200) and the response is "Success".
2. Send a POST request to the '/register' endpoint with the same username as in Step 1.
   - *Expected Result:* The request is unsuccessful (HTTP 401) and the response is "Failed".

Results:

- Step 1: The POST request was successful, and the response is "Success".
- Step 2: The POST request was unsuccessful, and the response is "Failed".

Status: PASSED

2. Login Endpoint

Objective: Verify the functionality of the '/login' endpoint to authenticate a user.

Steps:

1. Send a POST request to the '/login' endpoint with valid credentials for an existing user.

{ "username": "test_user", "password": "password123" }

   - *Expected Result:* The request is successful (HTTP 200), and the response contains an access token.
2. Send a POST request to the '/login' endpoint with invalid credentials.

{ "username": "test_user", "password": "wrong_password" }

- *Expected Result:* The request is unsuccessful (HTTP 401) and the response is "Failed".

Results:

- Step 1: The POST request was successful, and the response contains an access token.
- Step 2: The POST request was unsuccessful, and the response is "Failed".

Status: PASSED

3. Log Message to File Endpoint

Objective: Verify the functionality of the '/logfile' endpoint to log a message to a file.

Steps:

1. Send a POST request to the '/logfile' endpoint with a valid access token and the following payload:

{ "message": "Testing log message to file endpoint." }

- *Expected Result:* The request is successful (HTTP 200), and the response is "Logged message: [message]".

Results:

- Step 1: The POST request was successful, and the response is "Logged message: [message]".

Status: PASSED

4. Log Message to Database Endpoint

Objective: Verify the functionality of the '/logdb' endpoint to log a message to the database.

Steps:

1. Send a POST request to the '/logdb' endpoint with a valid access token and the following payload:

{ "app_id": 123, "message": "Testing log message to database endpoint.", "host_name": "localhost", "file": "app.py", "log_level": "info", "module": "main" }

- *Expected Result:* The request is successful (HTTP 200), and the response is "Logged message".

Results:

- Step 1: The POST request was successful, and the response is "Logged message".

Status: PASSED

5. Log Error Message to File Endpoint

Objective: Verify the functionality of the '/logerrorfile' endpoint to log an error message to a file.

Steps:

1. Send a POST request to the '/logerrorfile' endpoint with a valid access token and the following payload:

{ "message": "Testing log error message to file endpoint." }

- *Expected Result:* The request is successful (HTTP 200), and the response is "Logged message: [message]".

Results:

- Step 1: The POST request was successful, and the response is "Logged message: [message]".

Status: PASSED

6. Log Error Message to Database Endpoint

Objective: Verify the functionality of the '/logerrordb' endpoint to log an error message to the database.

Steps:

1. Send a POST request to the '/logerrordb' endpoint with a valid access token and the following payload:

{ "app_id": 123, "message": "Testing log error message to database endpoint.", "host_name": "localhost", "file": "app.py", "log_level": "error", "module": "main", "line": 42, "error_message": "Test error", "error_code": 500 }

- *Expected Result:* The request is successful (HTTP 200), and the response is "Logged message".

Results:

- Step 1: The POST request was successful, and the response is "Logged message".

Status: PASSED

7. Process Log Batch Endpoint

Objective: Verify the functionality of the '/logbatch' endpoint to process a batch of log messages.

Steps:

1. Send a POST request to the '/logbatch' endpoint with a valid access token and the following payload:

[{ "app_id": 123, "message": "Log message 1", "host_name": "localhost", "file": "app.py", "log_level": "info", "module": "main", "error": false, "db": true, "fmt_msg": { "message": "Formatted message 1" } }, { "app_id": 456, "message": "Error message 1", "host_name": "localhost", "file": "app.py", "log_level": "error", "module": "main", "line": 42, "error": true, "error_code" : 1001, "db": true, "fmt_msg": { "message": "Formatted error message 1" } }]

- *Expected Result:* The request is successful (HTTP 200), and all log messages are processed correctly.

Results:

- Step 1: The POST request was successful, and all log messages are processed correctly.

Status: PASSED

8. Check Consistency Endpoint

Objective: Verify the functionality of the '/check-consistency' endpoint to check the consistency between the file and database logs.

Steps:

1. Send a GET request to the '/check-consistency' endpoint with a valid access token.
    - *Expected Result:* The request is successful (HTTP 200), and the response indicates the consistency status (True or False).

Results:

- Step 1: The GET request was successful, and the response indicates the consistency status.

Status: PASSED

9. Make Consistent Endpoint

Objective: Verify the functionality of the '/make-consistent' endpoint to make the file and database logs consistent.

Steps:

1. Send a POST request to the '/make-consistent' endpoint with a valid access token.
    - *Expected Result:* The request is successful (HTTP 200), and the file and database logs are made consistent.

Results:

- Step 1: The GET request was successful, and the file and database logs are made consistent.

Status: PASSED

10. Get Log File Endpoint

Objective: Verify the functionality of the '/getfile' endpoint to retrieve log entries from the file.

Steps:

1. Send a GET request to the '/getfile' endpoint with a valid access token and optional query parameters to filter log entries.

- *Expected Result:* The request is successful (HTTP 200), and the response contains the log entries from the file.

Results:

- Step 1: The GET request was successful, and the response contains the log entries from the file.

Status: PASSED

11. Get Log Database Endpoint

Objective: Verify the functionality of the '/getdb' endpoint to retrieve log entries from the database.

Steps:

1. Send a GET request to the '/getdb' endpoint with a valid access token and optional query parameters to filter log entries.
    - *Expected Result:* The request is successful (HTTP 200), and the response contains the log entries from the database.

Results:

- Step 1: The GET request was successful, and the response contains the log entries from the database.

Status: PASSED

12. Delete Log File Endpoint

Objective: Verify the functionality of the '/deletefile' endpoint to delete log entries from the file.

Steps:

1. Send a POST request to the '/deletefile' endpoint with a valid access token and the following payload:

{ "message": "Log message 1" }

    - *Expected Result:* The request is successful (HTTP 200), and the specified log entries are deleted from the file.

Results:

- Step 1: The POST request was successful, and the specified log entries are deleted from the file.

Status: PASSED

13. Delete Log Database Endpoint

Objective: Verify the functionality of the '/deletedb' endpoint to delete log entries from the database.

Steps:

1. Send a POST request to the '/deletedb' endpoint with a valid access token and the following payload:

{ "message": "Log message 1" }

- *Expected Result:* The request is successful (HTTP 200), and the specified log entries are deleted from the database.

Results:

- Step 1: The POST request was successful, and the specified log entries are deleted from the database.

Status: PASSED