QA automation assignment

Test Report

Summary:

This document provides a detailed explanation of the QA automation assignment, highlighting the approach, methodology, and test results. The focus of the tests was to ensure the stability, security, and correctness of login functionality, deal retrieval, and related operations. Additionally, the assignment evaluated the functionality of posting, updating, and deleting comments on deals, as well as filtering forms associated with deals.

Objective

The primary objective of this assignment was to test the following functionalities:

- **User Login**: Verify successful login with valid credentials and appropriate error handling for invalid login attempts.
- **Authorization**: Ensure secure access to deal-related data using tokens.
- **Deal Operations**: Test operations like retrieving deals, filtering forms, posting, updating, and deleting comments.

Approach

The assignment followed a structured approach:

1. **Setup and Configuration**:
    - Python programming language was used.
    - *pytest* framework was chosen for test automation.
    - *requests* library was used for HTTP API calls.
2. **Test Case Design**:
    - Each functionality was broken down into individual test cases.
    - Both positive and negative test cases were created to ensure robustness.
    - Test cases were named descriptively for clarity and maintainability.
3. **Fixtures and Reusability**:

- Authentication token retrieval was implemented using fixtures to ensure reusability across tests.

4. **Error Handling**:
   - Tests included handling invalid inputs, missing tokens, and failed responses.

5. **Validation**:
   - Assertions were implemented to validate response status codes and payload contents.

Test Cases and Results:

1) Login Tests:

1. **Successful Login:**
   - Function: test_successful_login()
   - Description: Verifies login success using valid, registered credentials.
   - Expected Result: Status code 200, response contains "success" and a valid token.
   - Outcome: Login successfully authenticated; token returned in the response. ✅

2. **Login with Empty Credentials**
   - Function: test_empty_credentials()
   - Description: Verifies login failure when both email and password fields are empty.
   - Expected Result: Status code 400, response contains an error indicating invalid credentials.
   - Outcome: Login correctly rejected with appropriate error response. ✅

3. **Unregistered Email with Valid Password**
   - Function: test_unregistered_valid_email_with_existing_password()
   - Description: Verifies login failure when using a valid email format but an unregistered email address and a registered password.
   - Expected Result: Status code 400, response contains an error indicating incorrect credentials.
   - Outcome: Login attempt correctly rejected; error message matches expectations. ✅

4. **Invalid Email Format**
    - Function: test_invalid_email_format()
    - Description: Verifies login failure when using an invalid email format.
    - Expected Result: Status code 400, response contains an error indicating invalid credentials.
    - Outcome: Login attempt correctly rejected; error message indicates invalid credentials. ✅

5. **Invalid Password**
    - Function: test_invalid_password()
    - Description: Verifies login failure with an invalid password for a registered email address.
    - Expected Result: Status code 400, response contains an error indicating incorrect credentials.
    - Outcome: Login attempt correctly rejected; error message matches expectations. ✅

6. **Invalid Email and Password Combination**
    - Function: test_invalid_email_and_password_credentials()
    - Description: Verifies login failure when both email and password are invalid.
    - Expected Result: Status code 400, response contains an error indicating incorrect credentials.
    - Outcome: Login attempt correctly rejected; error message matches expectations. ✅

7. **Maximum Failed Login Attempts**
    - Function: test_max_failed_login_attempts()
    - Description: Verifies that multiple failed login attempts trigger a lockout response.
    - Expected Result: Status code 400, lockout message displayed after a certain number of attempts.
    - Outcome: Failed - The system did not trigger a lockout after the expected number of failed attempts (5). This needs further investigation to identify the root cause. ❌

2) **List of Deals Tests:**

8. **Verify List of Deals with Token**
    - Function: test_verify_list_deals() (uses get_auth_token fixture)
    - Description: Verifies the retrieval of deals using a valid authentication token provided by a fixture.
    - Expected Result: Status code 200, response contains exactly one deal titled "Shared deal for home assignment."
    - Outcome: Deals retrieved successfully with valid authentication. ✅

9. **Verify List of Deals without Token**
    - Function: test_verify_list_deals_without_token()
    - Description: Verifies that accessing deals without a token fails.
    - Expected Result: Status code 401, error response indicating authentication failure. And not getting a response.
    - Outcome: Failed - Received status code 200 without the expected response or error message. Additional investigation is required. (Using a try-except block reveals that no response is being returned) ❌

10. **Validate Answered Filter for Form**
    - Function: test_answered_sort(get_auth_token)
    - Description: Validates the "answered" filter functionality for retrieving forms associated with a specific deal.
    - Expected Result: Status code 200, response contains "data" field, and "message" field with value "Successful."
    - Outcome: Filter functionality works as expected. Status code 200 returned, response contains "data" field and "Successful" message. ✅

11. **Post New Comment on Deal**
    - Function: test_post_comment(get_auth_token)
    - Description: Verifies the ability to post a comment on a deal with the required details.
    - Expected Result: Status code 201, response contains "message" field with value "Successful," and "data" field with "text" matching the posted comment.

- Outcome: Comment posted successfully. Status code 201 returned, response contains "Successful" message and correct comment text. ✅

12. Update Existing Comment

- Function: test_update_comment()
- Description: Validates the functionality of updating an existing comment.
- Expected Result: Status code 200, response contains a "Successful" message, and the updated comment text matches the payload (the updated comment)
- Outcome: The comment was successfully updated with the expected details. ✅

13. Delete Existing Comment

- Function: test_delete_comment(post_comment_id, get_auth_token)
- Description: Validates the deletion of a specific comment using its unique ID.
- Expected Result: Status code 200, response contains an empty "data" field, and "message" field with value "Successful."
- Outcome: Comment deleted successfully. Status code 200 returned, response contains empty "data" field and "Successful" message. ✅

Below is a summary of test cases and their outcomes:

Test Case	Outcome
Successful Login	✅ Passed
Login with Empty Credentials	✅ Passed
Unregistered Email with Valid Password	✅ Passed
Invalid Email Format	✅ Passed
Invalid Password	✅ Passed
Invalid Email and Password Combination	✅ Passed
Maximum Failed Login Attempts	❌ Failed
Verify List of Deals with Token	✅ Passed
Verify List of Deals without Token	❌ Failed
Validate Answered Filter for Form	✅ Passed
Post New Comment on Deal	✅ Passed
Update Existing Comment	✅ Passed
Delete Existing Comment	✅ Passed

Conclusion:

This assignment demonstrated the importance of robust API testing to ensure system functionality, security, and reliability. While most functionalities performed as expected, the identified issues—such as the failure to trigger lockouts after multiple login attempts and the missing error message when accessing deals without a token—highlight areas that require further investigation.

To improve the system, the following actions are recommended:

- Review and address the lockout threshold for failed login attempts.

- Investigate and resolve the issue with accessing deals without an authentication token.

- Retest all features once the issues are resolved to ensure proper functionality.