**Test Execution Summary**

**Environment**:

* Python version: 3.12.7
* MySQL Database: flavorsacademy
* Test Framework: unittest

**Summary**:

* **Total Tests Run**: 12
* **Tests Passed**: 10
* **Tests Failed**: 2
* **Code Coverage**: (Provide percentage if available)

**Detailed Test Report**

**1. Test Name: test\_get\_course\_success**

* **Description**: This test checks if the API endpoint /api/courses/<id> correctly returns a 404 status code when the specified course ID does not exist.
* **Expected Outcome**: Status code 404 (Not Found).
* **Actual Outcome**: Status code 500 (Internal Server Error).
* **Failure Analysis**:
  + The backend returned a 500 error, which suggests an issue in the error-handling logic within the course retrieval endpoint.
  + This may be due to unhandled exceptions, incorrect parameter types, or database query failures when attempting to retrieve a non-existent course.
* **Resolution**:
  + **Action Items**:
    1. Review the /api/courses/<id> endpoint to ensure it correctly handles cases where the course ID does not exist.
    2. Add exception handling to return a 404 status code when no course is found.
    3. Test the endpoint with both valid and invalid IDs to verify the fix.

**2. Test Name: test\_login\_success**

* **Description**: Validates that the login endpoint returns a 200 status code when correct credentials are provided.
* **Expected Outcome**: Status code 200 (OK).
* **Actual Outcome**: Status code 401 (Unauthorized).
* **Failure Analysis**:
  + The returned 401 status code indicates unauthorized access, which may imply that the provided login credentials were not recognized.
  + This could result from test data setup issues or the absence of a proper record in the database.
* **Resolution**:
  + **Action Items**:
    1. Ensure that the student@example.com user account exists in the database with the expected credentials.
    2. Verify that the signup test for this user account was successful before running the login test.
    3. If login data persistence is an issue, set up database seeding or mock the login response during testing.

**Summary of Next Steps**

1. **Update Backend Logic**: Address error-handling logic in the /api/courses/<id> endpoint.
2. **Ensure Test Data Consistency**: Verify that all prerequisite data (such as user accounts) are correctly set up and accessible during each test run.
3. **Run Tests**: Re-run the tests after implementing fixes and document the updated results.

**Additional Notes**

* **Database Considerations**: Ensure the test environment cleans up any residual data between test runs to avoid conflicts, especially where unique constraints are involved.
* **Further Documentation**: Store this test documentation alongside the codebase, ideally in a TEST\_REPORT.md file, or integrate it with automated testing/reporting tools if available.

This structured approach provides clear insight into the test failures, expected behavior, analysis, and steps for resolution. It also serves as a good reference for future debugging and testing.