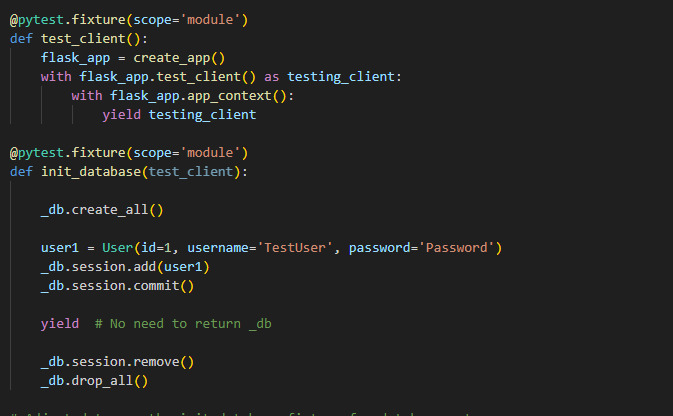
# Testing:

## Unit Testing

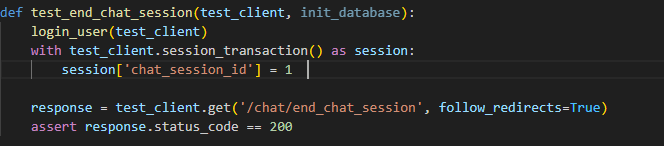
| **ID** | **Component** | **Test Case** | **Input** | **Expected Output** | **Success Criteria** |
| --- | --- | --- | --- | --- | --- |
| **TC-001** | **Start Chat Session (Authenticated)** | User is logged in and starts a chat session | User session with user\_id set | HTTP status code 201, creation of a new chat session | Test passes if the response status code is 201 |
| **TC-002** | **Start Chat Session (Unauthenticated)** | User is not logged in | No user\_id in session | HTTP redirect (302) or 401 Unauthorized | Test passes if the response is a redirect (302) or 401 |
| **TC-003** | **Get Chat Sessions (Authenticated)** | User is logged in and fetches chat sessions | User session with user\_id set | HTTP status code 200 and a list of chat sessions associated with the user | Test passes if the status code is 200 and the response contains the user's chat sessions |
| **TC-004** | **Get Chat Sessions (Unauthenticated)** | User is not logged in | No specific input needed | HTTP status code 200 | Test passes if the status code is 200 |
| **TC-005** | **Save Interaction (Authenticated)** | User is logged in and saves an interaction | User session with user\_id and chat\_session\_id set, JSON payload with user\_input and ai\_response | HTTP status code 200 with a success message | Test passes if the interaction is successfully saved and the response contains 'success' |
| **TC-006** | **Register User** | New user registration | POST data with username and password | HTTP status code 200 and the new user is found in the database | Test passes if a new user is created and can be found in the database |
| **TC-007** | **Login (Success)** | Existing user logs in successfully | POST data with valid username and password | HTTP status code 200 and session contains user\_id | Test passes if the user is logged in and session contains user\_id |
| **TC-008** | **Login (Failure)** | User fails to log in | POST data with invalid username and password | HTTP status code 200 with an 'Invalid username or password.' message | Test passes if the response contains the invalid login message |
| **TC-009** | **Submit (Authenticated)** | User submits input in a chat | JSON payload with user\_input, User session contains user\_id | HTTP status code 200 and the response JSON contains the same user\_input | Test passes if the response status is 200 and echoes user\_input |
| **TC-010** | **End Chat Session (Authenticated)** | User ends a chat session | User session with chat\_session\_id set | HTTP status code 200, indicating the chat session was successfully ended | Test passes if the chat session ends successfully |
| **TC-011** | **Fetch Chat Histories (Authenticated)** | User fetches their chat histories | User session contains user\_id | HTTP status code 200 and a JSON payload containing the user's chat histories | Test passes if the response contains the user's chat histories and status code is 200 |

## Designing the Unit Tests Using Pytest:

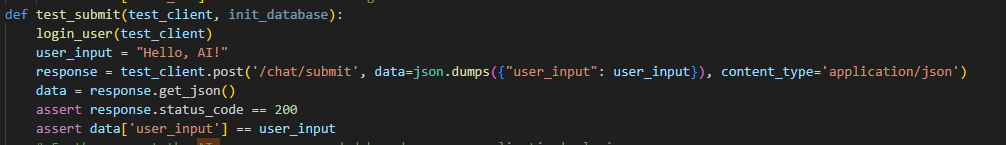
**Setting up the Test Environment and setting up the database before the Tests::**



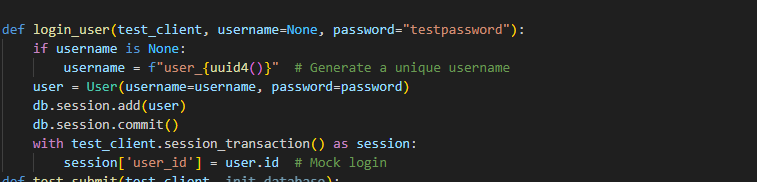
**Pytest for Ending Chat Session:**



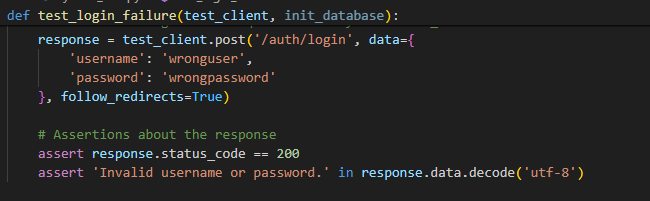
**Pytest for form submission and AI Response:**



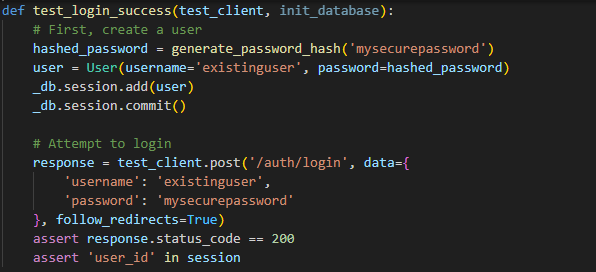
**Pytest Authentication/ Login:**



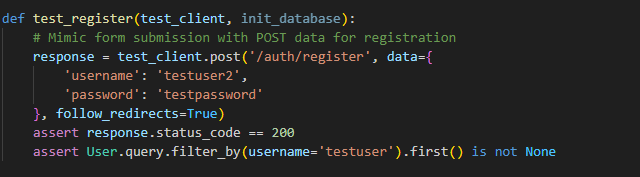
**Pytest Authentication/ Login Failure:**



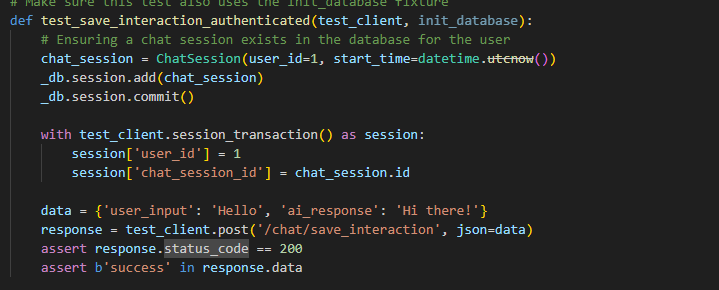
**Pytest Authentication/ Login Success:**



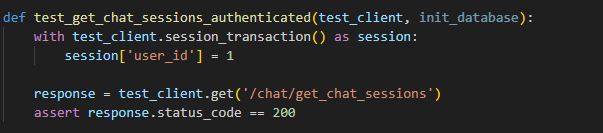
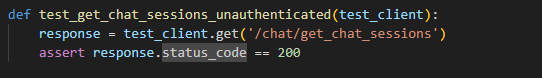
**Pytest Authentication/ Register:**



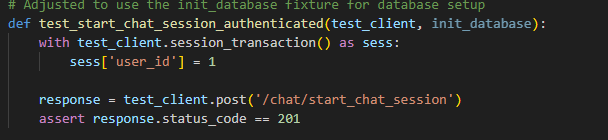
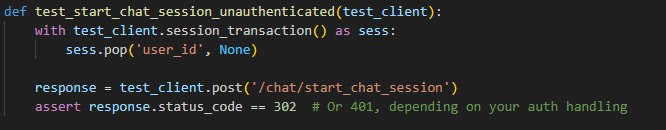
**Pytest Saving Interation of Users and ChatSession:**



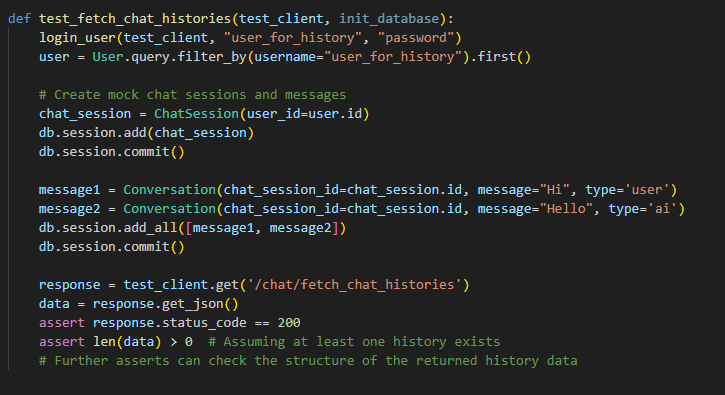
**Pytest to get chat sessions:**



**Pytest to start chat session:**



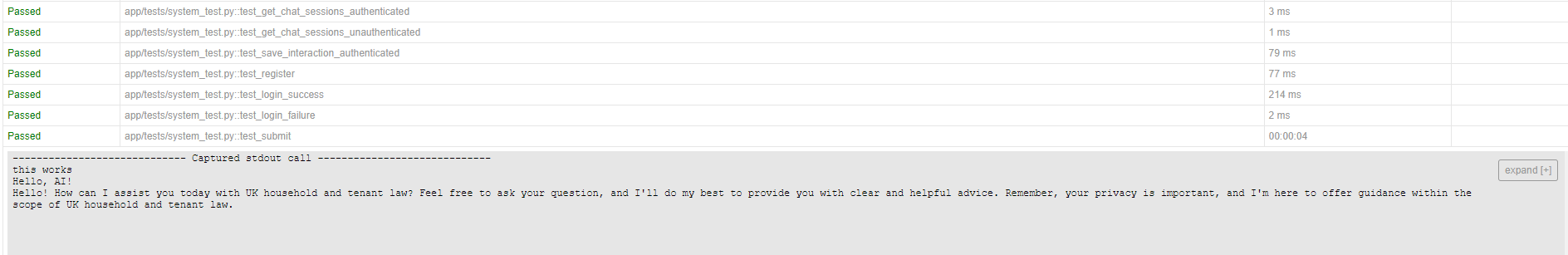
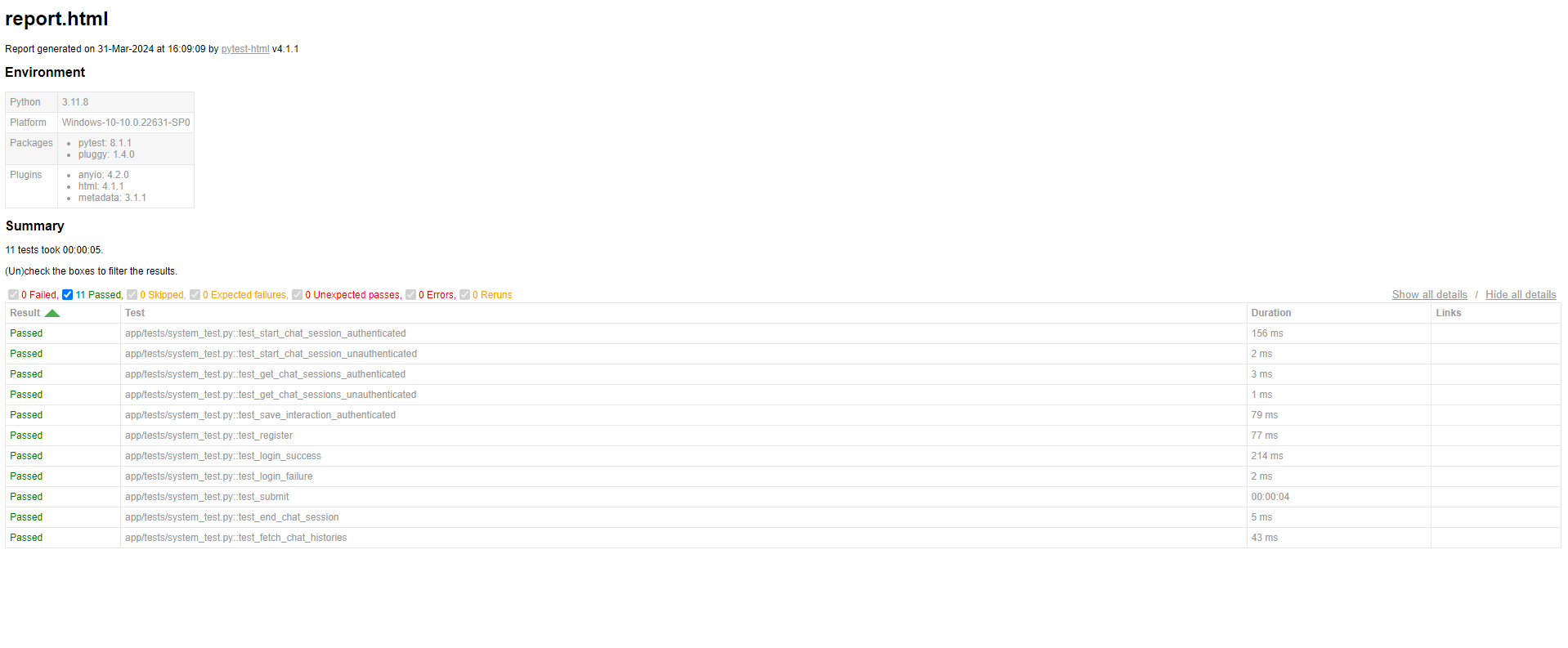
**Pytest get chat histories:**



## Unit Tests Outcome:

This chapter concludes the unit tests that were conducted for the Litigat8. The test results were all satisfactory with almost 80% complete success rate. With 2 test cases that resulted that resulted in warnings but they carried out the function they were supposed to. The two tests that resulted in warnings would be discussed in the next chapter.

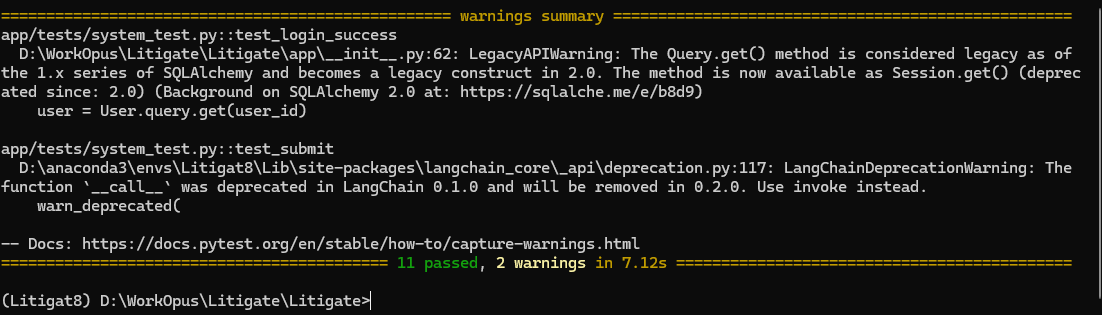
| **ID** | **Component** | **Test Case** | **Input** | **Expected Output** | **Success Criteria** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- |
| **TC-001** | Start Chat Session (Authenticated) | User is logged in and starts a chat session | User session with user\_id set | HTTP status code 201 creation of a new chat session | Test passes if the response status code is 201 | Pass |
| **TC-002** | Start Chat Session (Unauthenticated) | User is not logged in | No user\_id in session | HTTP redirect (302) or 401 Unauthorized | Test passes if the response is a redirect (302) or 401 | Pass |
| **TC-003** | Get Chat Sessions (Authenticated) | User is logged in and fetches chat sessions | User session with user\_id set | HTTP status code 200 and a list of chat sessions | Test passes if the status code is 200 and the response contains the user's chat sessions | Pass |
| **TC-004** | Get Chat Sessions (Unauthenticated) | User is not logged in | No specific input needed | HTTP status code 200 | Test passes if the status code is 200 | Pass |
| **TC-005** | Save Interaction (Authenticated) | User is logged in and saves an interaction | User session with user\_id and chat\_session\_id set JSON payload with user\_input and ai\_response | HTTP status code 200 with a success message | Test passes if the interaction is successfully saved and the response contains 'success' | Pass |
| **TC-006** | Register User | New user registration | POST data with username and password | HTTP status code 200 and the new user is found in the database | Test passes if a new user is created and can be found in the database | Pass |
| **TC-007** | Login (Success) | Existing user logs in successfully | POST data with valid username and password | HTTP status code 200 and session contains user\_id, indicating successful login | Test passes if the user is logged in and session contains user\_id | Pass |
| **TC-008** | Login (Failure) | User fails to log in | POST data with invalid username and password | HTTP status code 200 with an Invalid username or password message | Test passes if the response contains the invalid login message | Pass |
|  | Submit (Authenticated) | User submits input in a chat | JSON payload with user\_input, User session contains user\_id | HTTP status code 200 and the response JSON contains the same user\_input | Test passes if the response status is 200 and echoes user\_input | Pass |
|  | End Chat Session (Authenticated) | User ends a chat session | User session with chat\_session\_id set | HTTP status code 200, chat session was successfully ended | Test passes if the chat session ends successfully | Pass |
|  | Fetch Chat Histories (Authenticated) | User fetches their chat histories | User session contains user\_id | HTTP status code 200 and a JSON payload containing the user's chat histories | Test passes if the response contains the user's chat histories and status code is 200 | Pass |



## Unit Tests Outcome Discussion:

**Component Login(Success) :**

**func test\_login\_success:** The warning was originated from SQLALchemy. Which warned me about the Query.get() method has depreciated in the version 2.0 of the library. And instead advised me to use Session.get() instead to resolve the warning and any other compatibility issues in the future.

As a result, the login functions was updated to use the newer version of the function thus eliminating the warning for the test\_login\_success function

**Component Submit (Authenticated):**

**func test\_submit:**

The warning was originated from langchain\_core\_api warning about the deprecation of \_\_call\_\_ function within the version LangChain 0.1.0 and warning that it would be removed in 0.2.0. I wasn’t able to exactly pinpoint which method would be required to change as the error is not being originated from. But for the sake of this project and the version of LangChain that is being used for the project. It woudn’t effect the functioning of the project.

| **Integration Components** | **Test Case** | **Input** | **Expected Output** | **Success Criteria** |
| --- | --- | --- | --- | --- |
| Frontend <-> Backend API Calls | Test login API call | User login credentials | User authenticated / Session token | User can log in and start a session |
|  | Test query submission API call | Valid query | Query accepted / Advice returned | Query processed and advice returned |
| Backend <-> Database CRUD Operations | Test user creation in database through backend | New user details | User created in **Users** table | New user can log in |
|  | Test query processing and storage | Sample query | Query stored in UserQueries & NLPAnalysis | Query results and analysis are stored |
| NLP Engine <-> Database | Test storing NLP analysis results | Sample query | Results stored in NLPAnalysis | Analysis results retrievable for the query |
| Authentication <-> Database | Test authentication with database-stored credentials | User credentials | Authentication success/failure | Correct authentication against database data |
| Chat Interface <-> Chat Handling <-> Database | Test chat message flow | Chat message | Message stored in ChatHistories | Chat history updated with new message |

**Conclusion:**