CS667 Distributed Operating Systems

Spring 2019

Lab 2 Tests Document

Team Name: STIMA

Name: Aggrey Muhebwa
Github Handle: amuhebwa
Student ID: 32055729

Name: Zeal Shah
Github Handle: zealshah95
Student ID: 31737150

Summary

We have created three tests for the main functions of the front_end server.We chose writing functional tests over unit tests, because we want to test the functionality of the front_end server's interactivity with other servers over the network.

Where to Find the Script and Results?

In the file structure provided as part of the starter code, the folder for tests is on the same level as the src folder. However, when writing tests, the testing framework requires that the test file must be either on the same level or sub_level with the file that you are testing. As such, we put out test file, named *test_frontend.py* in the src folder along with other files (after discussions with the instructor). However, the output of this file has been left in the test folder.

How to Run the Tests?

To run the tests, make sure that *test_frontend.py* and *front_end.py* are in the same folder AND on the same machine. Additionally, make sure that all the other servers are running (see how to do this under **Implementation/Running the Program** in the design document).

Testing Setup

We import the file (*front_end.py*) that we want to test into *test_frontend.py* and then create a class reference that allows the in-build python module for testing to access all the functions of the *front_end.py* server. We then set up two functions for initiating and cleaning up the test suite once each of tests have been executed.
Three different test cases have been created-

1. **test_search_request**: This tests the *search_request* by directly making the API call on behalf of the original function. As in the original documentation, we search for a book topic called *graduate_school* and expect the response to be a JSON object that contains an HTTP code **200** to indicate that the request was successful and details on the two books under this topic; *Cooking for the Impatient Graduate Student and Xen and the Art of Surviving Graduate School.* In addition, the JSON object should also include the book numbers; *546 and 768.* In this test, we test that these exist, otherwise, if they do not exist or the HTTP code is not **200**, the tests must fail.

2. **test_lookup_request**: This tests the *lookup_request* functionality by making an API call on behalf of this function to search for a book with a given book number. As in the original function, it expects a JSON object as a return response, which we further decode and check if the query was successful. As the first check, the response should have the HTTP code **200** indicating the query was successful and in addition, it should contain the book number, corresponding to the item number that was requested. If it is different, the requests should fail.

3. **test_buy**: This tests the *buy* functionality by making a direct API call on behalf of this original function. The response should be a JSON object that has HTTP code **200** indicating that the request was successful and in addition, it should contain either "*Transaction successful.*" if the transaction was successful or "*Oops! We ran out of {book_number}'s stock. Please try again after sometime*". If we don't any of the two and the 200 HTTP response code, the tests should fail.

Conclusion

Our reason for focusing on testing the *front_end.py* server instead of other servers is based on the function that this server makes different calls to two servers *order.py* and *catalog.py* servers and is the only server that can be contacted by client. Therefore, the front-end server makes a good candidate for testing the correctness of store's distributed system operations. The three tests above demonstrate our understanding of creating functional tests as stated in the goals of the assignment. Additionally, performance testing for 1000 sequential requests and concurrent requests handling has been discussed in the performance evaluation document.