SE 3XA3: Test Report
Plagiarism Check

310, MXQ Squad
Wang Xu , wangx147
Mrinal Kumar Tiwari , tiwarim
Qifeng Xu and xuq14

April 7, 2020

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
April 6, 2020	1.0	Added the test report

1 Functional Requirements Evaluation

Description of Tests: These tests ensure that the system is able to do what it should do based on the functional requirements. These tests will test for 4 corresponding functional requirements in the SRS.

Test Name: FRE1

Results: The users get a status code of 200 and a message of success when they register with a new username and password.

Test Name: FRE2

Results: When the users try to register with the same username twice, there is an error message displayed.

Test Name: FRE3

Results: The user enters two texts for plagiarism checking and gets a output of similarity ratio.

Test Name: FRE4

Results: After refilling the tokens, the present amount of tokens is updated to the sum of current amount of tokens and token refill amount.

2 Nonfunctional Requirements Evaluation

2.1 Look and Feel

Test Name: NFR1

Test: Refer to the software requirement specification document, We go through the website to see if it has two text boxes for input, if there is a suitable background colour, if the user could understand what to do next, if there is a numerical results after user submitted the text.

Results: The product displays two text boxes and some buttons on the website. It chooses pink and purple as its background colour. It has the information telling the user to register the account, check the text and refill the tokens. At the same time, it displays a numerical result after user has submitted the text.

2.2 Usability and Humanity

Test Name: NFR2-1

Test: We have input user name '123' and password '123' to check if it registered successfully.

Results: The account registered successfully.
Test Name: NFR2-2

Test: We have input two texts with 100 percent similarity to see if the results match the one we expected.

Results: The result of check test is 100 percent similarity.
Test Name: NFR2-3

Test: We input admin password 'Admiral123' and 5 tokens to see if tokens are top up.

Results: The tokens top up by amount of 5.

2.3 Robustness

Test Name: NFR3-1

Test: We have input two Chinese texts to see if it can check the language other than English.

Results: It can check the similarity ratio for Chinese texts as well.
Test Name: NFR3-2

Test: We have input two emoji to see if it can check emoji other than English.

Results: It can check the similarity ratio for emoji as well.

2.4 Performance

Test Name: NFR4

Test: We tested the similarity check part to see if the result can be displayed in 0.1 second.

Results: All the tests matched what we expected. The result is displayed within less than 0.1 second after the user submits the texts.

2.5 Operational and Environmental

Test Name: NFR5

Test: Our website is hosted by AWS and we try to run it on different devices, for example, laptop, cellphone and iPad.

Results: The product is hosted by a website for local use before it is sold. The product is able to run on many different devices as long as the devices can open web browsers.

2.6 Maintainability and Support

Test Name: NFR6

Test: We did a survey test that all the user could run the program on their local machines.

Results: The potential users could run the program on their local machines and the product is universally accessible by users.

2.7 Security

Test Name: NFR7

Test: We did a survey test that all the user could access the product on the website.

Results: All the public visitor is able to access the product on the website. The product does not alter its source code and prevent incorrect data input.

2.8 Cultural and Political

Test Name: NFR8

Test: It can be clearly seen that our product is available in English.

Results: The product is not offensive to religious or ethnic groups. The product is available in English.

2.9 Legal

Test Name: NFR9

Test: Basically it cannot be tested, but we are sure that it does not compromise any laws.

Results: The product does not compromise any laws

3 System Testing

3.1 Tests for Functional Requirements

Test Name	FRE1
Initial State	The interface asks user to input a new username and password.
Input	Valid username and password.
Expected Output	The system returns a status code of 200 and a message of registration completed.
Actual Output	A status code of 200 and a message of ”you successfully signed up for the api”.

Table 2: Test for FRE1

Test Name	FRE2
Initial State	The interface asks user to input a new username and password.
Input	An existing username and password.
Expected Output	The system returns an error message of registration failed and the username already exists.
Actual Output	A status code of 301 and an error messages ”username already exists”.

Table 3: Test for FRE2

Test Name	FRE3
Initial State	The interface asks user to input two texts.
Input	Two valid texts.
Expected Output	The system returns a similarity ratio.
Actual Output	The similarity ratio of two texts.

Table 4: Test for FRE3

Test Name	FRE4
Initial State	The interface asks user to enter the amount of tokens.
Input	Valid integer.
Expected Output	The system returns a message of refill successfully and the current token amount is updated.
Actual Output	A message of ”Tokens refilled successfully”, the current token amount is updated correctly.

Table 5: Test for FRE4

3.2 Tests for Nonfunctional Requirements

Test Name	NFR2-1
Initial State	The interface asks user to input a new username and password.
Input	A unique username which is ”123” and a password which is ”123”.
Expected Output	The system returns a message of registered successfully.
Actual Output	The system returns a status code of 200 and a message of ”you successfully signed up for the api”.

Table 6: Test for NFR2-1

Test Name	NFR2-2
Initial State	The interface asks user to input two texts.
Input	Two valid texts with a proven similarity ratio of 100 percent.
Expected Output	The system returns a result of 100 percent similarity.
Actual Output	A similarity ratio of 100 percent.

Table 7: Test for NFR2-2

Test Name	NFR2-3
Initial State	The interface asks user to input admin password and token amount, the current token amount is 6.
Input	A valid username of ”123” and admin password which is ”Admiral123” and 5 tokens.
Expected Output	The system returns a message of refill successfully and updates current token amount.
Actual Output	A message of ”Tokens refilled successfully”, and the current token amount is updated to 11.

Table 8: Test for NFR2-3

Test Name	NFR3-1
Initial State	The interface asks user to input two texts.
Input	Two identical texts with Chinese characters.
Expected Output	The system returns a status code of 200 and a similarity ratio of 100.
Actual Output	The system returns a status code of 200 and a similarity ratio of 100.

Table 9: Test for NFR3-1

Test Name	NFR3-2
Initial State	The interface asks user to input two texts.
Input	Two identical emoji.
Expected Output	The system returns a status code of 200 and a similarity ratio of 100.
Actual Output	The system returns a status code of 200 and a similarity ratio of 100.

Table 10: Test for NFR3-2

Test Name	NFR4
Initial State	The interface asks user to input two texts.
Input	The interface asks user to submit the request and the timer is set to 0.
Expected Output	The system returns a result in less than 0.1 second.
Actual Output	The calculated processing time is 0.00729s.

Table 11: Test for NFR4

Test Name	NFR5
Initial State	Preparation of multiple devices that are able to connect to network and have a valid browser.
Input	Open the webpage on different devices and submit the request.
Expected Output	The system works and returns expected results.
Actual Output	The system works and returns expected results.

Table 12: Test for NFR5

4 Changes Due to Testing

4.1 Tests for Functional Requirements

There is no change due to testing for functional requirements, since all results of the completed tests matched with their expected output.

4.2 Tests for Nonfunctional Requirements

During the tests, we noticed the processing time of each function is much less than 1 second. it is less than 0.01 second in most of the cases, therefore, we decided to change the time limit in the nonfunctional requirement of performance to 0.1 second.

5 Automated Testing

5.1 Tests for Functional Requirements

The automated unit tests are applied for 4 functional requirements that are able to be tested through pytest. Each tested functional requirements has at least a regular case, a failed case case and a succeed case with unusual inputs.

For the FRE1, 3 test cases are created. There is one with unique and valid username and password that passes the test, one with the username that already exists that fails the test and one with the username consist of 3 special symbols that passes the test.

For the FRE2, it is similar to the failed test case in FRE1. There are only 2 test cases for it because there is no abnormal inputs that can be used to test for it.

For the FRE3, 4 test cases are created. There is one with valid username, password and texts that passes the test, one with username that does not exist which fails the test, one with invalid password that passes the test with abnormal status code of 302 and one with a username that run out of tokens passes the test with abnormal status code of 303.

For the FRE4, 4 test cases are created. There is one with valid username and admin password that passes the test, one with non-exist username that passes the test with a abnormal status code of 301, one with valid username and invalid admin password that passes the test with abnormal status code of 304 and one with wrong amount of tokens that fails the test.

5.2 Tests for Nonfunctional Requirements

Each test case for Nonfunctional Requirement is created based on the test name, initial state, input, and expected output in the section of system testing. As a result, all the actual outputs from the test cases are identical to the actual outputs in the table of system testing.

6 Trace to Requirements

Test	Requirements
Functional Requirements Testing	
FRE1	FR1
FRE2	FR2,FR3
FRE3	FR4,FR5
FRE4	FR7
Non-functional Requirements Testing	
NFR1	NF3.1.1,NF3.1.2
NFR2-1	NF3.2.1-NF3.2.4
NFR2-2	NF3.2.1-NF3.2.4
NFR2-3	NF3.2.1-NF3.2.4
NFR3-1	NF3.3.4
NFR3-2	NF3.3.4
NFR4	NF3.3.1-NF3.3.3
NFR5	NF3.4.1,NF3.4.2
NFR6	NF3.5.1, NF3.5.2
NFR7	NF3.6.1, NF3.6.2
NFR8	NF3.7
NFR9	NF3.8.1

Table 13: Trace Between Tests and Requirements

7 Trace to Modules

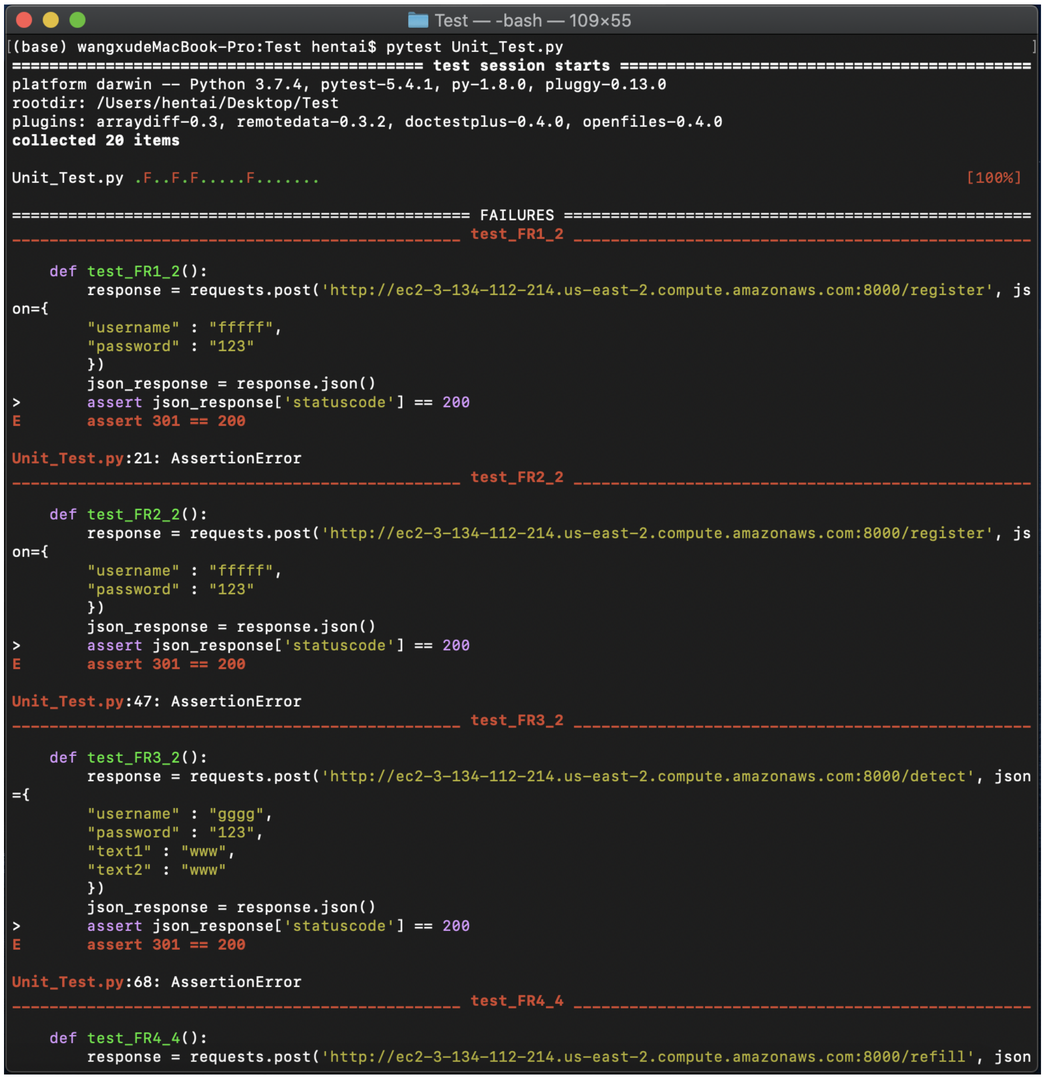
Test	Modules
Functional Requirements Testing	
FRE1	M2
FRE2	M2
FRE3	M3
FRE4	M4
Non-functional Requirements Testing	
NFR1	M1,M5
NFR2-1	M2
NFR2-2	M3
NFR2-3	M4
NFR3-1	M1,M3,M5,M6,M7
NFR3-2	M1,M3,M5,M6,M7
NFR4	M1,M3,M5,M6,M7
NFR5	M1,M2,M3,M4,M5,M6,M7
NFR6	M1,M2,M3,M4,M5,M6,M7
NFR7	M1,M2,M3,M4,M5,M6,M7
NFR8	M1,M2,M3,M4,M5,M6,M7
NFR9	M1,M2,M3,M4,M5,M6,M7

Table 14: Trace Between Tests and Modules

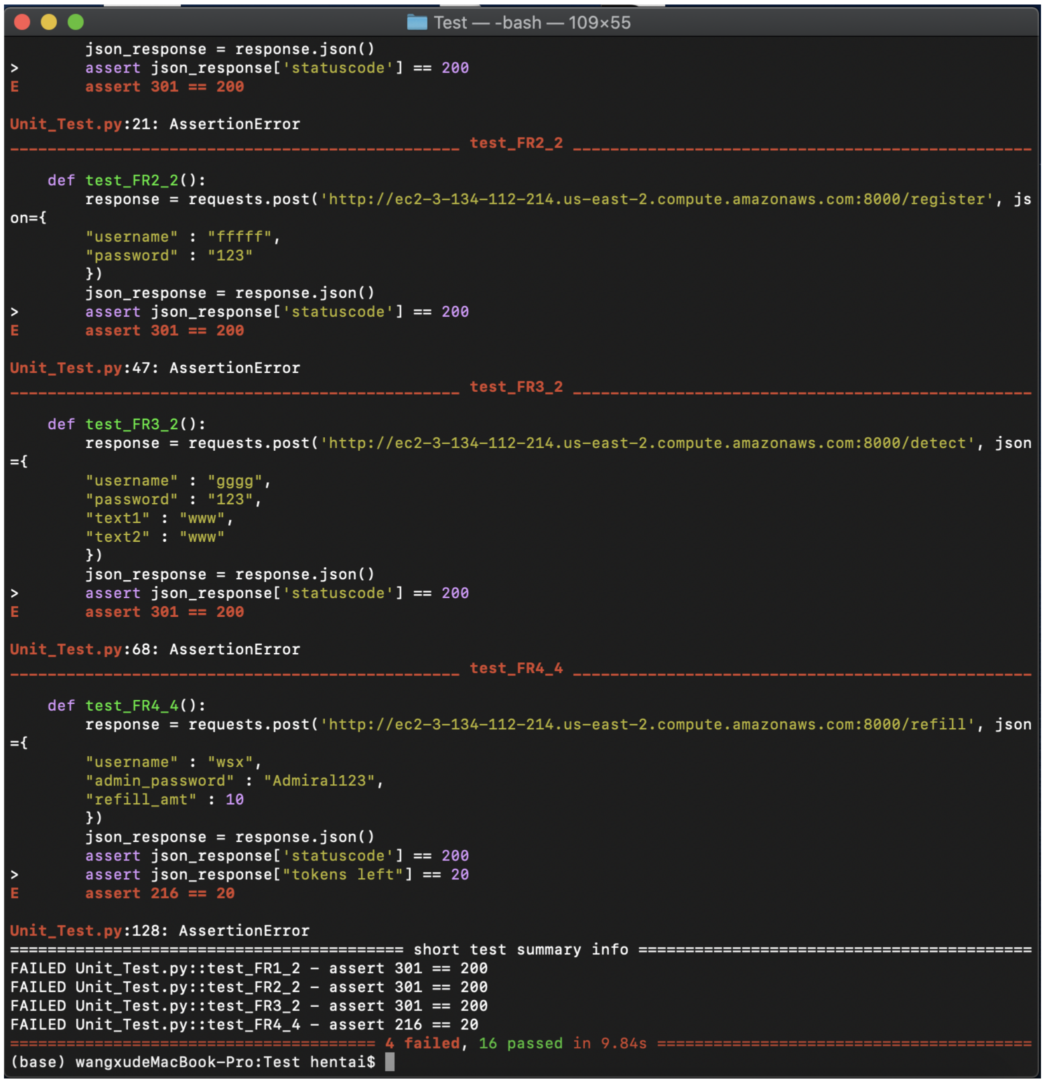
8 Code Coverage Metrics

We has managed to produce roughly 100 percent statement coverage and 90 percent condition coverage through our tests. All of the modules and all the different situations of status code have been covered through our tests. It can be seen from the trace to modules section, we covered each module multiple times.

9 Appendix



```
[(base) wangxudeMacBook-Pro:Test hentai$ pytest Unit_Test.py                                                 ]
============================================ test session starts ============================================
platform darwin -- Python 3.7.4, pytest-5.4.1, py-1.8.0, pluggy-0.13.0
rootdir: /Users/hentai/Desktop/Test
plugins: arraydiff-0.3, remotedata-0.3.2, doctestplus-0.4.0, openfiles-0.4.0
collected 20 items

Unit_Test.py .F..F.F.....F.......                                                                      [100%]

================================================= FAILURES ==================================================
________________________________________________ test_FR1_2 _________________________________________________

    def test_FR1_2():
        response = requests.post('http://ec2-3-134-112-214.us-east-2.compute.amazonaws.com:8000/register', js
on={
        "username" : "fffff",
        "password" : "123"
        })
        json_response = response.json()
>       assert json_response['statuscode'] == 200
E       assert 301 == 200

Unit_Test.py:21: AssertionError
________________________________________________ test_FR2_2 _________________________________________________

    def test_FR2_2():
        response = requests.post('http://ec2-3-134-112-214.us-east-2.compute.amazonaws.com:8000/register', js
on={
        "username" : "fffff",
        "password" : "123"
        })
        json_response = response.json()
>       assert json_response['statuscode'] == 200
E       assert 301 == 200

Unit_Test.py:47: AssertionError
________________________________________________ test_FR3_2 _________________________________________________

    def test_FR3_2():
        response = requests.post('http://ec2-3-134-112-214.us-east-2.compute.amazonaws.com:8000/detect', json
={
        "username" : "gggg",
        "password" : "123",
        "text1" : "www",
        "text2" : "www"
        })
        json_response = response.json()
>       assert json_response['statuscode'] == 200
E       assert 301 == 200

Unit_Test.py:68: AssertionError
________________________________________________ test_FR4_4 _________________________________________________

    def test_FR4_4():
        response = requests.post('http://ec2-3-134-112-214.us-east-2.compute.amazonaws.com:8000/refill', json
```



Test — -bash — 109×55

```
        json_response = response.json()
>       assert json_response['statuscode'] == 200
E       assert 301 == 200

Unit_Test.py:21: AssertionError
________________________________________________ test_FR2_2 ________________________________________________

    def test_FR2_2():
        response = requests.post('http://ec2-3-134-112-214.us-east-2.compute.amazonaws.com:8000/register', js
on={
        "username" : "fffff",
        "password" : "123"
        })
        json_response = response.json()
>       assert json_response['statuscode'] == 200
E       assert 301 == 200

Unit_Test.py:47: AssertionError
________________________________________________ test_FR3_2 ________________________________________________

    def test_FR3_2():
        response = requests.post('http://ec2-3-134-112-214.us-east-2.compute.amazonaws.com:8000/detect', json
={
        "username" : "gggg",
        "password" : "123",
        "text1" : "www",
        "text2" : "www"
        })
        json_response = response.json()
>       assert json_response['statuscode'] == 200
E       assert 301 == 200

Unit_Test.py:68: AssertionError
________________________________________________ test_FR4_4 ________________________________________________

    def test_FR4_4():
        response = requests.post('http://ec2-3-134-112-214.us-east-2.compute.amazonaws.com:8000/refill', json
={
        "username" : "wsx",
        "admin_password" : "Admiral123",
        "refill_amt" : 10
        })
        json_response = response.json()
        assert json_response['statuscode'] == 200
>       assert json_response["tokens left"] == 20
E       assert 216 == 20

Unit_Test.py:128: AssertionError
========================================== short test summary info ==========================================
FAILED Unit_Test.py::test_FR1_2 - assert 301 == 200
FAILED Unit_Test.py::test_FR2_2 - assert 301 == 200
FAILED Unit_Test.py::test_FR3_2 - assert 301 == 200
FAILED Unit_Test.py::test_FR4_4 - assert 216 == 20
======================================= 4 failed, 16 passed in 9.84s ========================================
(base) wangxudeMacBook-Pro:Test hentai$
```