**CS673 Software Engineering**

**Team 2 - Communication Tool   
Software Test Document (STD)**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Laura Kocubinski | Team Leader | *Laura Kocubinski* | 10/30/19 |
| Hang Shi | Backup Team Leader  Requirement Leader |  |  |
| Xi You | QA Leader |  |  |
| Jhuanderson Macias | Design and Implementation Leader |  |  |
| Jhuanderson Macias | Security Leader |  |  |
| Bofeng (Beven) Liu | Configuration Leader | *Bofeng Liu* | 10/31/19 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| 0.1 | Hang Shi | 10/31/2019 | First Draft |
|  |  |  |  |

# Introduction

This document outlines the testing done on our Django chat web application.

# Tools

* Python 3.6 or higher
* Django 2.2.5
* Django Channels 2.0.2 or higher
* Selenium - tool for automating web application testing
* Chrome Browser
* ChromeDriver - WebDriver for Chrome which tests websites (Windows/Mac/Linux)

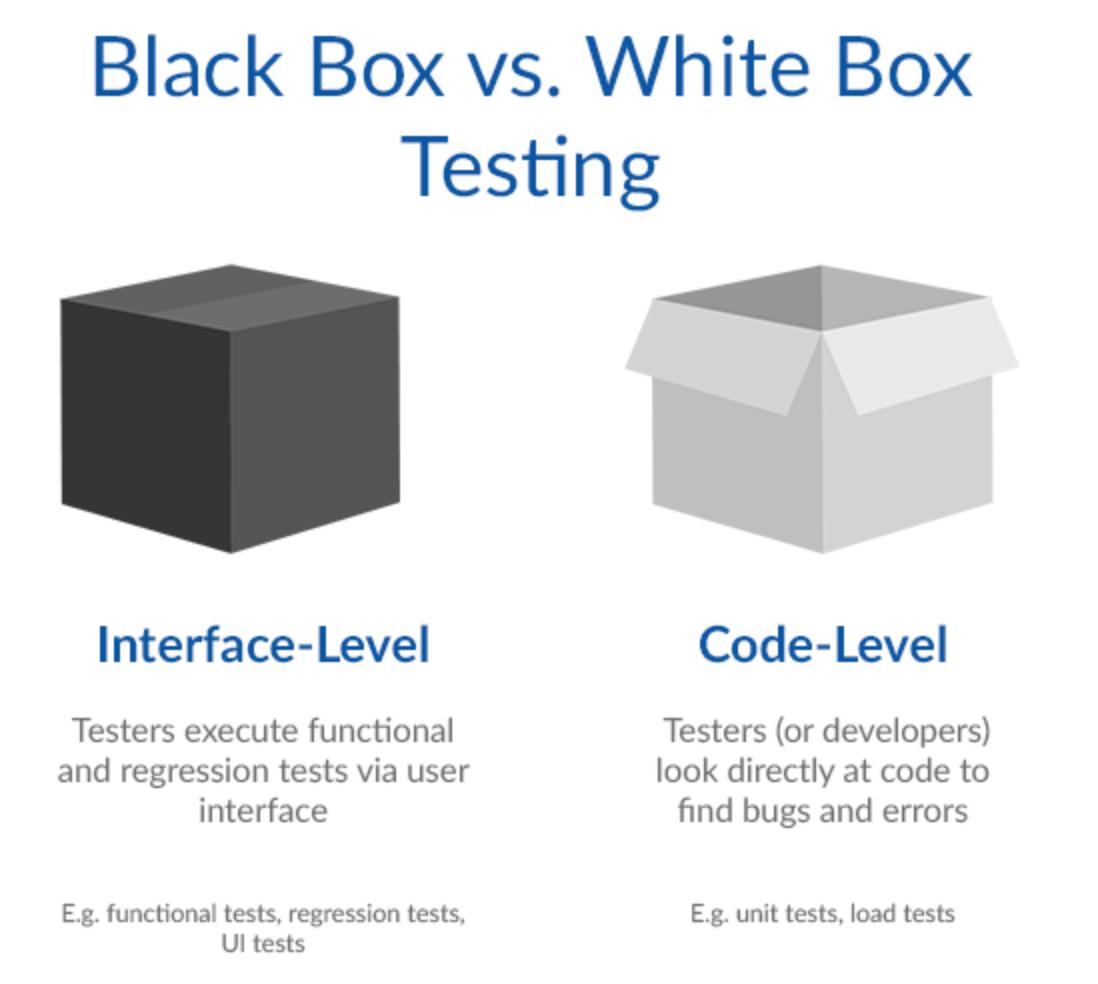
# Test Summary

## Unit Tests

### unittest

The Python unit testing framework, sometimes referred to as “PyUnit,” is a Python language version of JUnit. unittest is the de facto standard unit testing framework for Python.

## System Tests



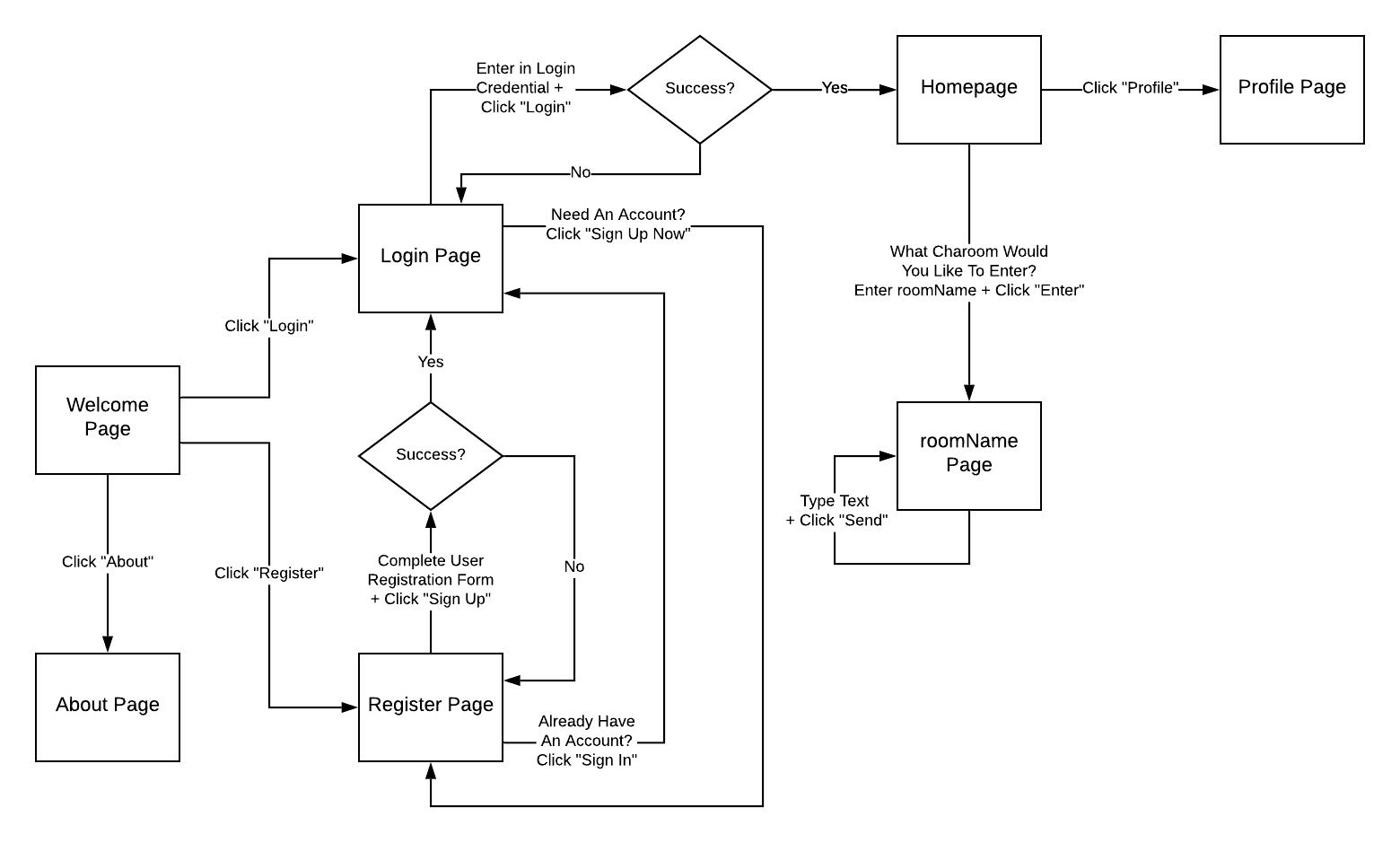
### Integration Test

### Regression Test

## Acceptance Tests

### UI Tests

All UI Black-Boxing testing should follow the step of UI flow diagram.



## Testing Metrics

## Test Design Efficiency

## 

## Cost of Finding a Defect

## 

## Passed Test Cases (Percentage)

## 

## Coverage Defect Rate

# Test Report

In this section, we give a detailed description of each test case performed and the result.

## Django Forms

### UserCreationForm (Registration)

This tests the UserCreationForm, which handles user registration. This is located in users/tests.py

|  |
| --- |
| *Test Items*   * test\_is\_form\_valid   + Check if form is valid with username, password1, and password2 fields * test\_form\_fields   + Check that username, password1, and password2 fields are stored as expected * test\_weak\_password   + Check that form throws an error with a weak password * test\_form\_with\_email\_field   + Check that form still works with email field added   *Test Priority*:Low; register functionality is working  *Dependencies:* UserCreationForm  *Preconditions:*None  *Input Data:*   * UserCreationForm * Dictionary with input fields as keys   *Test Steps:* From communication\_app root, run from terminal $ python manage.py test users *Post-Conditions:* N/A  *Expected Output:* Depends on test; form.is\_valid() returns True or False  *Actual Output:* As expected  *Pass or Fail:* Pass  *Bug ID/GitHub Link:* N/A  *Additional Notes:* N/A |

### Automation (Selenium)

|  |
| --- |
| *# chat/tests.py*  **from** **channels.testing** **import** ChannelsLiveServerTestCase  **from** **selenium** **import** webdriver  **from** **selenium.webdriver.common.action\_chains** **import** ActionChains  **from** **selenium.webdriver.support.wait** **import** WebDriverWait  **class** **ChatTests**(ChannelsLiveServerTestCase):  serve\_static = **True** *# emulate StaticLiveServerTestCase*  **@classmethod**  **def** setUpClass(cls):  super().setUpClass()  **try**:  *# NOTE: Requires "chromedriver" binary to be installed in $PATH*  cls.driver = webdriver.Chrome()  **except**:  super().tearDownClass()  **raise**  **@classmethod**  **def** tearDownClass(cls):  cls.driver.quit()  super().tearDownClass()  **def** test\_when\_chat\_message\_posted\_then\_seen\_by\_everyone\_in\_same\_room(self):  **try**:  self.\_enter\_chat\_room('room\_1')  self.\_open\_new\_window()  self.\_enter\_chat\_room('room\_1')  self.\_switch\_to\_window(0)  self.\_post\_message('hello')  WebDriverWait(self.driver, 2).until(**lambda** \_:  'hello' **in** self.\_chat\_log\_value,  'Message was not received by window 1 from window 1')  self.\_switch\_to\_window(1)  WebDriverWait(self.driver, 2).until(**lambda** \_:  'hello' **in** self.\_chat\_log\_value,  'Message was not received by window 2 from window 1')  **finally**:  self.\_close\_all\_new\_windows()  **def** test\_when\_chat\_message\_posted\_then\_not\_seen\_by\_anyone\_in\_different\_room(self):  **try**:  self.\_enter\_chat\_room('room\_1')  self.\_open\_new\_window()  self.\_enter\_chat\_room('room\_2')  self.\_switch\_to\_window(0)  self.\_post\_message('hello')  WebDriverWait(self.driver, 2).until(**lambda** \_:  'hello' **in** self.\_chat\_log\_value,  'Message was not received by window 1 from window 1')  self.\_switch\_to\_window(1)  self.\_post\_message('world')  WebDriverWait(self.driver, 2).until(**lambda** \_:  'world' **in** self.\_chat\_log\_value,  'Message was not received by window 2 from window 2')  self.assertTrue('hello' **not** **in** self.\_chat\_log\_value,  'Message was improperly received by window 2 from window 1')  **finally**:  self.\_close\_all\_new\_windows()  *# === Utility ===*  **def** \_enter\_chat\_room(self, room\_name):  self.driver.get(self.live\_server\_url + '/chat/')  ActionChains(self.driver).send\_keys(room\_name + '**\n**').perform()  WebDriverWait(self.driver, 2).until(**lambda** \_:  room\_name **in** self.driver.current\_url)  **def** \_open\_new\_window(self):  self.driver.execute\_script('window.open("about:blank", "\_blank");')  self.driver.switch\_to\_window(self.driver.window\_handles[-1])  **def** \_close\_all\_new\_windows(self):  **while** len(self.driver.window\_handles) > 1:  self.driver.switch\_to\_window(self.driver.window\_handles[-1])  self.driver.execute\_script('window.close();')  **if** len(self.driver.window\_handles) == 1:  self.driver.switch\_to\_window(self.driver.window\_handles[0])  **def** \_switch\_to\_window(self, window\_index):  self.driver.switch\_to\_window(self.driver.window\_handles[window\_index])  **def** \_post\_message(self, message):  ActionChains(self.driver).send\_keys(message + '**\n**').perform()  **@property**  **def** \_chat\_log\_value(self):  **return** self.driver.find\_element\_by\_css\_selector('#chat-log').get\_property('value') |

### Open New Window

|  |
| --- |
|  |

### Enter Chatroom

|  |
| --- |
|  |

* 1. Switch to window

|  |
| --- |
|  |

* 1. Post message

|  |
| --- |
|  |

* 1. Close all new window

|  |
| --- |
|  |

* 1. Change Font Size

|  |
| --- |
|  |

* 1. Create new chatting room

|  |
| --- |
| *Test name: create new chatting room*  *Test items: create new chatting room button, chat room input area, “log in” button~~, cancel button, save button~~*  *Test priority: high*  *Preconditions: user log into system successfully, create new room button*  *input data:*  *· chat room input area*  *Test steps:*  *~~· user log in system successfully~~*  *~~· click create new room button~~*  *~~· type new room name in textbox~~*  *~~click save button~~*  *~~click create new room button~~*  *~~· type new room name in textbox~~*  *~~click cancel button~~*  *Postcondition: users log in successfully.*  *Expected output:*  *~~· after clicking the save button, new chat room name will appear in the group name llistbox.~~*  *~~and when you click the new room label, a new chatting room window appears.~~*  *~~after clicking the cancel button, new chat room name will become non-exist.~~*  *Actual output: same with expected*  *Pass or Fail: pass* |

* 1. Set and display status.

|  |
| --- |
| *Test name: Set and display status*  *Test items: create new teams button, team name input area, “log in” button, user list*  *Test priority: medium*  *Preconditions: login to the system, have users online/offline*  *input data:*  *· user list*  *different user account*  *Test steps:*  *~~· create two users, one user log into system.~~*  *~~· check the user list area~~*  *~~· log out this user account~~*  *~~· check the user list area~~*  *Postcondition: two users can be successfully created and log in /log out successfully.*  *Expected output:*  *~~· after one user log in, the name color becomes blue.~~*  *~~after log out, the name color become grey.~~*  *~~when there is no users, the user list is empty.~~*  *Actual output: same with expected*  *Pass or Fail: pass* |

* 1. Reset password via email

|  |
| --- |
|  |

* 1. Display when users joins and leaves a chatroom

|  |
| --- |
|  |

* 1. Change text size

|  |
| --- |
|  |

* 1. Change font styles

|  |
| --- |
|  |

* 1. Display when users joins and

|  |
| --- |
|  |

*n1\*.Welcome Page button function testing (front-end)*

|  |
| --- |
| *Test name: Welcome Page button function testing*  *Test items: Click the several different buttons in Welcome Pages, find out which button can not provide the function to go to HTML front-end pages.*  *Test priority: medium*  *Preconditions: non login requirement, just enter the main welcome pages*  *input data:*  *· non directly input data*  *Test steps:*  *~~· 1.Click the ‘about’ button to join the about pages, check and return back.~~*  *~~· 2.Click the sub-button (login) to join the login page, check and return back~~*  *~~· 3. Click the sub-button (register) to join the login page, check.~~*  *~~· check the user list area~~*  *Postcondition: any users can be successfully from the welcome page to join the other html static page and see the relevant info successfully.*  *Expected output:*  *~~· 1.User can successful see the ‘about page.html’ and see the relevant no error information.~~*  *~~2. User can successful see the ‘login page.html’ and see the template of login and the other requirement data.~~*  *~~3. User can successful see the ‘register page.html’ and see the template of register requirement and other no-error data.~~*  *Actual output: same with expected*  *Pass or Fail: pass* |

*n2\*.ETC button function testing (front-end)*

|  |
| --- |
| *Test name: Etc button function testing*  *Test items: Random clicks the several different buttons in different static html pages, find out which button or the relevant static writing error or the bootstrap.js library have the non-link error, which can not show the UI design for the front-end.*  *Test priority: medium*  *Preconditions: non login requirement, just click the random button to join the different static html page.*  *input data:*  *· non directly input data*  *Test steps:\**  *~~· Click the ‘random post’ picture to check will that zoom it or nor.~~*  *~~· Click the sub-button (post-relevant) to join the post page, check show the template or not~~*  *~~· Click the sub-button (contact) to join the contact page, check if input the contact information or not.~~*  *~~· check the user list area~~*  *Postcondition: any users (login or not) can be successfully from the welcome page to join the other html static page and see the relevant info successfully.*  *Expected output:*  *~~· User can successful zoom the picture larger to see clearly.~~*  *~~User can successful see the ‘login page.html’ and see the static post time-line and the other requirement data.~~*  *~~User can successful see the ‘contact page.html’ and see the template of contact sheet requirement and other no-error data.~~*  *Actual output: same with expected*  *Pass or Fail: pass* |

* 1. *Welcome Page button function testing (front-end)*

|  |
| --- |
| *Test name: Welcome Page button function testing*  *Test items: Click the several different buttons in Welcome Pages, find out which button can not provide the function to go to HTML front-end pages.*  *Test priority: medium*  *Preconditions: non login requirement, just enter the main welcome pages*  *input data:*  *· non directly input data*  *Test steps:*  *~~· 1.Click the ‘about’ button to join the about pages, check and return back.~~*  *~~· 2.Click the sub-button (login) to join the login page, check and return back~~*  *~~· 3. Click the sub-button (register) to join the login page, check.~~*  *~~· check the user list area~~*  *Postcondition: any users can be successfully from the welcome page to join the other html static page and see the relevant info successfully.*  *Expected output:*  *~~· 1.User can successful see the ‘about page.html’ and see the relevant no error information.~~*  *~~2. User can successful see the ‘login page.html’ and see the template of login and the other requirement data.~~*  *~~3. User can successful see the ‘register page.html’ and see the template of register requirement and other no-error data.~~*  *Actual output: same with expected*  *Pass or Fail: pass* |

* 1. Etc button function testing (front-end)

|  |
| --- |
| *Test name: Etc button function testing*  *Test items: Random clicks the several different buttons in different static html pages, find out which button or the relevant static writing error or the bootstrap.js library have the non-link error, which can not show the UI design for the front-end.*  *Test priority: medium*  *Preconditions: non login requirement, just click the random button to join the different static html page.*  *input data:*  *· non directly input data*  *Test steps:\**  *~~· Click the ‘random post’ picture to check will that zoom it or nor.~~*  *~~· Click the sub-button (post-relevant) to join the post page, check show the template or not~~*  *~~· Click the sub-button (contact) to join the contact page, check if input the contact information or not.~~*  *~~· check the user list area~~*  *Postcondition: any users (login or not) can be successfully from the welcome page to join the other html static page and see the relevant info successfully.*  *Expected output:*  *~~· User can successful zoom the picture larger to see clearly.~~*  *~~User can successful see the ‘login page.html’ and see the static post time-line and the other requirement data.~~*  *~~User can successful see the ‘contact page.html’ and see the template of contact sheet requirement and other no-error data.~~*  *Actual output: same with expected*  *Pass or Fail: pass* |

# References

1. unittest — Unit testing framework. Python 3.8.0 Documentation. <https://docs.python.org/2/library/unittest.html>.
2. <https://github.com/django/django/blob/master/tests/auth_tests/test_forms.py>

# Glossary

*Will we do the last 3 testing?*

Testing Cases *(editing)*

n.\*UI testing process

All UI Black-Boxing testing should follow the step of UI flow diagram.