CITS5501

Quality Assurance Report

Test and Automation Project

Damon van der Linde 21506136

18th May 2018

Table of Contents

Acronyms and Abbreviations

ODTS	Overdue task status
SCDA	Set completion date automatically
CSF	Comment snippet format
LTCA	List, task, creation by admin
CMTD	Comment and mark task done
RLA	Remove list by admin

List of Figures

1. Introduction

This report will focus on a provided web project known as CITS5501-Todo and will describe test cases used to test its functionality which can be located within test.py that will supplied alongside this document. The basic concept behind the provided web application is to allow a system administrator to create a task list which can be added to and edited by the users to keep track of ongoing issues or unimplemented features for a certain product or service. As the requirements of the project wasn't explicitly provided, a series of assumptions based on observations on the system interactions and source code will be explored and used in the formulation of the unit and browser tests. Descriptions and scenarios for each of the developed tests will be provided featuring some form of rationale behind implementing these. The report will then proceed to cover a basic plan of which involves describing how these implemented tests could be expanded on in the future to provide for better test coverage. Finally there will also be a section which details any bugs found during testing alongside a description of how these issues should be examined, and tracked in an instance where the project may still be in the development stage.

1.1. Assumptions about project requirements and functionality

- No other user can create or modify lists other than the administrator user
- Users of any privilege level can add, remove or edit tasks within a list

2. Description of tests

2.1. Unit tests

2.1.1. ODTS

When a task is created in the system a due date is specified. The method being examined with this unit test, compares the system date to that of the due date to determine where or not the particular task is overdue. The purpose of this test is to ensure that the task overdue status method behaves correctly such that if the date is in the past the function should yield a true result, otherwise should yield a false result.

2.1.2. SCDA

When a task is marked as done, the completed date is automatically assigned to the current system date at that particular moment. The purpose of this unit test is to test that the function behaves correctly by marking a created task as done and comparing its completion date to that of the current system's date.

2.1.3. CSF

The administrator users of the system has a comment snippet feature where by in the list_display associated with them, the name of the author behind the latest comment as well as 35 characters of that individuals comment is provided. The purpose of this test is to ensure that the format of this snippet is correctly presented.

2.2. Browser based tests

2.2.1. LTCA

The first browser based test involves simulating a situation where by the system administrator logs into the system, creates a new list, adds a task to the newly added list and then proceeds to sign out of his/her account. This browser based test firstly checks that system handles the login of an administrator correctly through examining the details of the homepage such as checking the heading on the page as well as the name of the user at the bottom. Next, the test evaluates the system in terms of correctly displaying the add a list page after clicking the create new list button. After filling in the name of the list and hitting submit it examines destination of the user to ensure that the the correct page was being provided. Alongside this, examination of a visual cue provided at the top of the screen below the navigation bar indicating that the new list has been added is conducted. At this stage, the test navigates to the newly added list and clicks on it. On the page corresponding to the list clicked on, the test finds the add task button and clicks on it. As it is a drop down form the test waits until the elements are visible before proceeding to fill in the task name, description and due date. The due date field has some discrepancies between different operating systems on the tested browser (chrome) which affect the interaction method with the element.

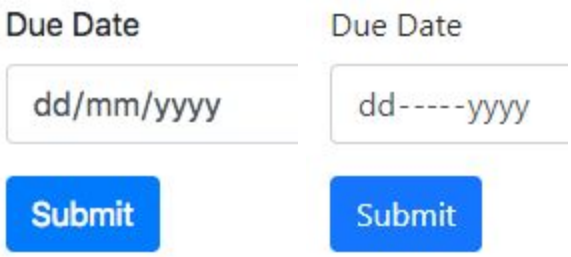

Figure 1 - Mac chrome due date (left) and Windows chrome due date (right)

The test first checks the OS name as this will be use in determining the interaction method. If the detected system is not windows then the due date information is formatted as a string in the form of ddmmyy. If the detected system is windows then the date info is separated into two strings of the nature ddmm and yy. Between these two string the tab key is also sent. Afterwards the submit button is pressed and the page heading is checked to ensure the user arrived at the correct place. Once the browser is back on the the task list page. It tests for a

success alert being displayed at the top indicating that the new task has been added. Finally the system performs the logout request and proceeds on to the next test.

2.2.2. CMTD

The second browser based test involved simulating a scenario where a regular user logs into the system, proceeds to clicking on one of the lists on the homepage followed by one of the tasks associated with it, leaving a comment on it and then marking the task as done while ending the session through logging off. This particular browser based test involves checking that a regular user is logged in properly through checking the heading on the page and the user name at the bottom. Next step of the testing involves the user navigating to one of the lists' main page where it showcases all the tasks. Again the heading of the page is evaluated to ensure the user is on the right page. The number of incomplete tasks are calculated by counting the number of rows within the tables and subtracting one from that to account for he headings in the first row. The next stages involves clicking on one of the incomplete tasks that are available ensuring the user ends up on the task details page which corresponds to the task they wanted to examine. After this has been verified the test then examines the comment system by filling in the comment text area with some text after which the add comment button is pressed. To ensure the the page has been updated to reflect the addition of a comment, the heading of comment section is examined. The test then proceeds to mark the task as done through pressing the mark done button by which afterwards it examines the page name and the alert message that appears to ensure it matches the required details. At this point the number of incomplete task is also calculated and compared to the previous number to ensure that the task has been removed from the incomplete list. Finally the test proceeds to log the user off.

2.2.3. RLA

The third browser test simulates a scenario where an administrator logs into the system and proceeds to remove a list from the system. The system checks that the admin users has indeed been correctly logged in through checking the page title as well as the username at the bottom. The next phase of the test involves clicking on one of the lists provided and then pressing on the delete button. The test then checks that a warning page is displayed indicating to the end user ensuring that this is the correct action they want to execute checking the title of the webpage. The it proceeds to pressed the delete button again and observing the alert status displayed afterwards at the top of the browser confirming that the list which was clicked on was in fact the list that was removed from the system. Finally the system then logs out the user.

3. Rationale for tests

3.1. Unit tests

3.1.1. Rationale for OTDS

This test was chosen as the overdue_status method isn't one that is provided by some library rather it is a custom method constructed for a particular purpose related to this project and as such requires validation.

3.1.2. Rationale for SCDA

The aforementioned test was selected as this is a backend custom method devised to record completion details of tasks automatically when they are marked as done instead of the user filling in this information. This is one of the basic functionalities of the system and since it isn't apart of system libraries it required validation.

3.1.3. Rationale for CSF

The reason behind implementing this test was due to it being a custom method that relates to a particular "hidden" part of the system which isn't necessarily viewed extensively. Also being that it is a custom method that was implement, validation was required for its functionality.

3.2. Browser based tests

3.2.1. Rationale for LTCA

This test validates that the system handles login of administrator user correctly as well as ensure some of the functionality associated with a user of such privilege status such as list creation works correctly alongside its visual cues. Finally it also checks that other basic functionality such as task creation and logging off.

3.2.2. Rationale for CMDT

This test validate that basic functionality such as commenting on task and marking a test as done is handled correctly and that the necessary visual cues are displayed to indicate to the user that their request was handled successfully.

3.2.3. Rationale for RLA

This test examines and validate that admin functionality such as removing a list works correctly as well as being promoted with the necessary warnings before deleting the list and then correctly displaying visual cues after the deletion task has been completed.

4. Test Coverage

4.1. Unit tests

The main metric that was considered for analysing test coverage of the unittests of the project involved measuring the number of functions that have been tested compared to the overall number of custom (non library specific) functions available. A number of methods were identified totaling to around 14 which should be covered however due to the specification of the project and the time constraints imposed only 3 of these functions were tested resulting in a test coverage of 21.43%.

4.2. Browser based tests

The metric chosen to analyse the coverage of the browser based test involves considering the various operations and comparing the number of operations evaluated to that of the overall number of operations available. For this project around 16 operations could be conducted which consisted of the following: login successfully, login failed, view list, add task, view completed tasks, delete list, mark task done, create list, view task, add comment, edit task, delete task, in-list link, view my tasks, search tasks and logout. For this project the devised scenarios covered 9 out of the 16 operations yielding a test coverage of 56.25%.

5. Plan for extending tests

Improvement of the test coverage in regards to both the unit tests and the browser based tests can be achieved through the addition of more tests to cover the remainder of the functions and or operations that were not covered during the current implementation. In addition to expanding the test coverage a function can be written that will allow for the browser based tests to be examined on a vast array of different browsers to ensure compatibility. The productivity of the tests can also be improved through reduction in execution time by handling tests in parallel. Another area of consideration is different operating system environments mixed in with the different browsers on each platform.

6. Bugs and errors during execution

6.1. Inconsistent return values for overdue status

During the execution of the first unit test, it was discovered that the overdue status function returns none instead of false if the due date has not passed. None is meaningless and the function should be altered to return false instead. This form of bug can simply be logged and tracked in a version control software whereby a developer can be assigned to rectifying the issue.

6.2. Undefined method within the comment system

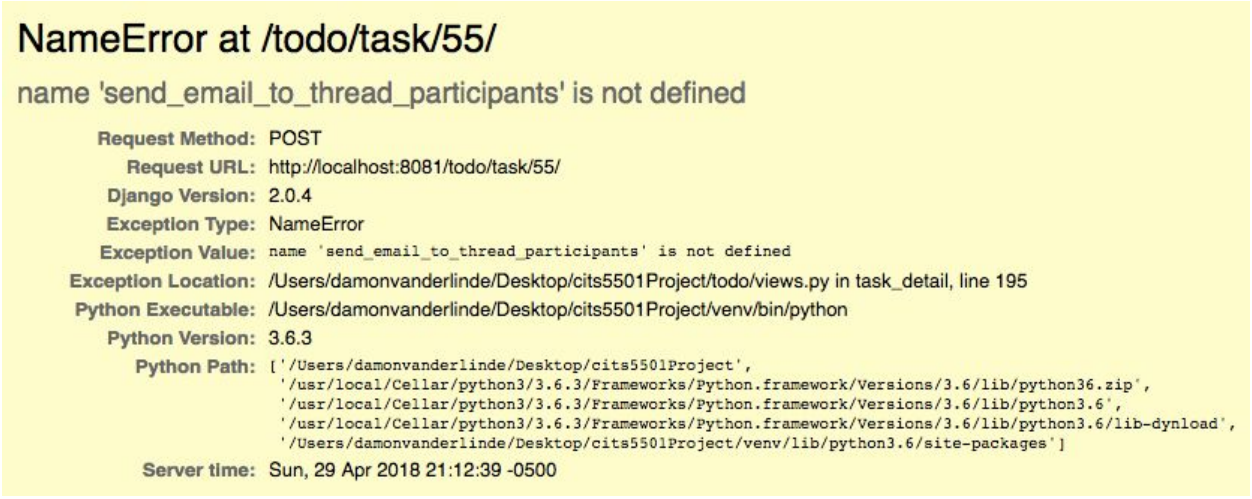

Figure 2 - Comment system bug

During the second selenium test scenario where the comment system is tested it was observed that there was an issue. The comment system experienced a an undefined method call. The name of the particular method indicates that it encompasses the notion that subscriber of that particular thread should be notify by the addition of the comment through email. This type of bug can be addressed through logging the issue in a version control software such as github, assigning priority levels to outstanding issues and distributing these amongst developers to resolve.