

# Content Testing

## COS 301 ASSIGNMENT CONTENT TESTING TEAM

Duran Cole (13329414)  
Kyhle Ohlinger (11131952)  
Andreas du Preez (12207871)  
Aluwani Simetsi (11322935)  
Michelle Swanepoel (13066294)  
Zander Boshoff (12035671)  
Tebogo Seshibe (13181442)  
Kgomotso Sito (12243273)  
Thabang Letageng (13057937)

April 2015

### Github Links

Content Testing  
Team A files  
Team B files

version: 1.0

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Threads Integration</b>	<b>1</b>
2.1	Threads Team A . . . . .	1
2.1.1	Functional Testing . . . . .	1
2.1.2	Non-Functional Testing . . . . .	2
2.2	Threads Team B . . . . .	2
2.2.1	Functional Testing . . . . .	2
2.2.2	Non-Functional Testing . . . . .	3
<b>3</b>	<b>Quality Requirements</b>	<b>3</b>
<b>4</b>	<b>Maintainability</b>	<b>3</b>
<b>5</b>	<b>Scalability</b>	<b>3</b>
<b>6</b>	<b>Performance Requirements</b>	<b>3</b>
<b>7</b>	<b>Reliability and Availability</b>	<b>4</b>
<b>8</b>	<b>Security</b>	<b>4</b>
<b>9</b>	<b>Auditability</b>	<b>4</b>
<b>10</b>	<b>Testability</b>	<b>4</b>
<b>11</b>	<b>Usability</b>	<b>4</b>
<b>12</b>	<b>Integrability</b>	<b>5</b>
<b>13</b>	<b>Deployability</b>	<b>5</b>
13.1	Threads Comparison . . . . .	5
<b>14</b>	<b>Status Integration</b>	<b>6</b>
14.1	Status Team A . . . . .	6
14.1.1	Functional Testing . . . . .	6
14.1.2	Non-Functional Testing . . . . .	7
14.2	Status Team B . . . . .	7

14.2.1	Functional Testing . . . . .	7
14.2.2	Non-Functional Testing . . . . .	9
14.3	Status Comparison . . . . .	9
<b>15</b>	<b>Resources Integration</b>	<b>10</b>
15.1	Resources Team A . . . . .	10
15.1.1	Functional Testing . . . . .	10
15.1.2	Non-Functional Testing . . . . .	11
15.2	Resources Team B . . . . .	12
15.2.1	Functional Testing . . . . .	12
15.2.2	Non-Functional Testing . . . . .	14
15.3	Resources Comparison . . . . .	14
<b>16</b>	<b>Reporting Integration</b>	<b>15</b>
16.1	Reporting Team A . . . . .	15
16.1.1	Functional Testing . . . . .	15
16.1.2	Non-Functional Testing . . . . .	16
16.2	Reporting Team B . . . . .	17
16.2.1	Functional Testing . . . . .	17
16.2.2	Non-Functional Testing . . . . .	18
16.3	Reporting Comparison . . . . .	18

# 1 Introduction

The purpose of this document is to provide incite into what was produced by both Content Team A (Buzz++) and Content Team B (D3). Groups were to provide an infrastructure to their low level teams and then build integration tests that would then allow the functional teams produced code to be tested and eventually added into the system. Tests were done per sub module (e.g. reporting, resource, threads and status) as to break up the integration that was to be done and provide an easier way to compare the teams work and functional code.

## 2 Threads Integration

The tests provided below show all cases in which the Threads module was to make use of another Content module and the expected and actual results of each tests. We then explain which non-functional requirements have not been fulfilled in the overall Threads module.

### 2.1 Threads Team A

#### 2.1.1 Functional Testing

##### Submit a Post

**Expected Test output** To Submit a Post the following needs to be intergrated:

- User's status points needs to be incremented (Status)
- The total number of posts for that thread needs to be increased (Reporting)
- The total number of active members in a thread needs to be increased (if user is new in thread) (Reporting)
- Should be able to add a resource to a post (Resources)

**Actual Test output** This test was a failure for the following reasons

- Intergration team didn't use threads functional team's code(only Resources and Reporting are added).

### 2.1.2 Non-Functional Testing

**Problem name 1 identified - e.g. Scalability** this is why this is an issue with scalability, blah blah blah

**Example of problem** this problem is caused because of potatoes. maybe add picture if required.

**Problem name 2 identified - e.g. Security** this is why this is an issue with Security, blah blah blah

**Example of problem** this problem is caused because of potatoes. maybe add picture if required.

## 2.2 Threads Team B

### 2.2.1 Functional Testing

#### Submit a Post

**Expected Test output** To Submit a Post the following needs to be intergrated:

- User's status points needs to be incremented (Status)
- The total number of posts for that thread needs to be increased (Reporting)
- The total number of active members in a thread needs to be increased (if user is new in thread) (Reporting)
- Should be able to add a resource to a post (Resources)

**Actual Test output** This test was a failure for the following reasons

- Intergration team didn't use threads functional team's code(only Resources and Reporting are added).
- No integration between the other modules (status, reporting or resources).

### 2.2.2 Non-Functional Testing

## 3 Quality Requirements

This section looks st the manner in which the code is made and maintained

## 4 Maintainability

- How flexible and extensible is the code and, can the system be easily maintained in future?
- The developers implemented their system very simplistically and provided comments with explanations of their code. The comments provided contain expected parameters, expected results and, potential throws.

## 5 Scalability

- The system should be easily scalable to large systems from the University of Pretoria to even larger systems.
- Due to it's simplistic nature and implementation, the system could easily be scaled.

## 6 Performance Requirements

- Reporting operations should take no more than 5 seconds while non-reporting operations should take less than (or equal to) 0.2 seconds.

- This system does not work so we cannot test.

## **7 Reliability and Availability**

- Should support fail-safe over safety of components and deployment without a single point of failure.
- The system itself completely failed as there were multiple errors.

## **8 Security**

- Check authentication against LDAP and flexible configurable authorization framework.
- No authorization is handled.

## **9 Auditability**

- The systems logs all requests and responses as stringified JSON object.
- They printed out whenever a function was called but not kept in a database.

## **10 Testability**

- Must be testable through unit testing and integration tests. Should verify that pre and post conditions met.
- Unit tests have been implemented but they do not succeed as the system itself is flawed.

## **11 Usability**

- The system should be intuitive and fun to use.
- This section is directed more towards the final end-product.

## 12 Integrability

- The system should be able to easily integrate future requirements.
- The simplistic nature allows for this.

## 13 Deployability

- The system must be deployable on Linux servers, different database persistence environments and, different repositories for authentication credentials.
- The system is deployable on Linux server.

### 13.1 Threads Comparison

According to the above tests both teams failed to integrate Threads with in the overall system, as all covered tests were failures and did not provide the expected functionality for the reasons given above.



## 14 Status Integration

The tests provided below show all cases in which the Status module was to make use of another Content module and the expected and actual results of each tests. We then explain which non-functional requirements have not been fulfilled in the overall Status module.

### 14.1 Status Team A

#### 14.1.1 Functional Testing

**NB: Problem** This module came with a read me file which gave instructions detailing how this module should be run. Following the instructions in the read me file, the module failed to run on a Linux machine.

#### Requirement 1 - Get Status For Profile

**Expected Functionality** This test must provide the ability to assess a number of measures around individual Buzz Space contributions.

**Actual Test Output** This test was a failure due to the above mentioned problem.

#### Requirement 2 - Calculate Access Level

**Expected Functionality** Be able to use the newly created ways of calculating a status for a profile on a Buzz Space.

**Actual Test Output** This test was a failure due to the above mentioned problem.

#### Requirement 3 - Assess Profile

**Expected Functionality** The user will be shown the profile they wish to access.

**Actual test output** This use case was a success.

## **Requirement 4 - Create Appraisal Type**

**Expected Functionality** User should be able to create an appraisal type, which includes the upload of appraisal images done by resources.

**Actual test output** This use case was a success.

## **Requirement 5 - Assign Appraisal To Post**

**Expected Functionality** Appraisal type should be assigned to the post(s).

**Actual test output** This use case was a success.

**Test Coverage Analysis** Although hundred percent of the use cases were implemented successfully, the overall test was a failure because the of the above mentioned problem.

### **14.1.2 Non-Functional Testing**

**Problem** The module failed to run using the given instructions in the provided readme file of the module.

**Security** 3 This requirement could not be tested due to the above mentioned problem. 4

## **14.2 Status Team B**

### **14.2.1 Functional Testing**

#### **Requirement 1 - Get Status For Profile**

**Expected Functionality** This test must provide the ability to assess a number of measures around individual Buzz Space contributions.

**Actual Test output** This test was successful as it provides ways to create new ways in which a profile status is calculated for a BuzzSpace. The only problem was that the needed authorisation was not implemented where it must have been. Only lecturers are supposed to be able to create these new ways.

## **Requirement 2 - Calculate Access Level**

**Expected Functionality** Be able to use the newly created ways of calculating a status for a profile on a Buzz Space.

**Actual Test output** This function was implemented successfully, but the specific Buzz Space for which this new calculation method must added was never specified.

## **Requirement 3 - Assess Profile**

**Expected Functionality** The user will be shown the profile they wish to access.

**Actual Test output** Not a success because of the following contract services being denied:

- No authorisation was done, thus all users could add a new way of calculating a status and not lecturers only.

## **Requirement 4 - Create Appraisal Type**

**Expected Functionality** User should be able to create an appraisal type, which includes the upload of appraisal images done by resources.

**Actual Test Output** Not a success because of the following contract services being denied:

- The specific Buzz Space for which the status calculation method must be changed was never specified or used to change the method.

## **Requirement 5 - Assign Appraisal To Post**

**Expected Functionality** Appraisal type should be assigned to the post(s).

**Actual test Output** This use case was not implemented in the file from the final commit.

## **Requirement 6 - Activate Appraisal Type**

**Expected Functionality** Activate the appraisal type for grading.

**Actual Test Output** This use case was not implemented in the file from the final commit.

**Test Coverage Analysis** 33 percent of the use cases were implemented successfully

### **14.2.2 Non-Functional Testing**

**Security** The authorisation in terms of who can change and create calculation of status methods was never done.

**Example of problem** For example, if a person wanted to change the way a status is calculated, every member of a Buzz Space will be able to do it. Only the lecturers are supposed to be able to do this.

## **14.3 Status Comparison**

Team A managed to implement 100 percent of the functions they had to do where as Team B only managed to implement a third of the functionality. Overall however both tests failed in a sense because of the failure to run on Linux that Team A presented and the lack of functionality by Team B.

## 15 Resources Integration

The tests provided below show all cases in which the Resources module was to make use of another Content module and the expected and actual results of each tests. We then explain which non-functional requirements have not been fulfilled in the overall Resources module.

### 15.1 Resources Team A

#### 15.1.1 Functional Testing

##### Requirement 1 - Upload Resources

**Expected Test output** Add the uploaded resource to the list of resources, and display the list

**Actual Test output** Option to upload available, but not functioning (does not add resource to the list)

- Because the team didn't integrate this, implemented the function

##### Requirement 2 - Remove Resource

**Expected Test output** Remove resource and display list of resources with resource of interest removed

**Actual Test output** Removed resource and displayed list of resources with resource of interest removed

##### Requirement 3 - Add Constraint

**Expected Test output** Add constraint with specified details and display list of constraints with the constraint of added

**Actual Test output** Added constraint with specified details and display list of constraints with the constraints of added included

##### Requirement 4 - Remove Constraint

**Expected Test output** Remove constraint and display list of constraints with constraint of interest removed

**Actual Test output** Removed constraint and displayed list of constraints with constraint of interest removed

### **Requirement 5 - Update Constraint**

**Expected Test output** Change constraint details to those specified in the field (size) and display list of constraints with the updated constraint included

**Actual Test output** Changed constraint details to those specified in the field (size) and displayed list of constraints with the updated constraint included

### **15.1.2 Non-Functional Testing**

**Security** Data can be manipulated by unauthorized parties, that is removal of resources and/or constraints

**Example of problem** Lack of authentication and authorization, to confirm if a party is authorized to make changes to resources

**Responsive / Performance** Takes time to load pages, and response to most queries also take time

**Example of problem** Long server response time, lack of browser caching

**Usability** Users get impatient (website takes long to respond), and understanding overview of site is difficult (especially when working with resources)

**Example of problem** Feedback is not provided when users must wait, and website value and purpose was not provided

**Quality** Error messages are not provided and/or error are not fixed

**Example of problem** Lack of Identification and rectification of faults

## 15.2 Resources Team B

### 15.2.1 Functional Testing

#### Requirement 1 - Add Resource Type

**Test 1 input** Type = image/jpeg max size = 10000000

**Expected Test output** The new mime type is expected to be added to the database to allow new data types to be uploaded by the client later.

**Actual Test output** This test was a success.

**Test 2 input** Type = test max size = 10000000

**Expected Test output** Invalid mime type exception

**Actual Test output** The invalid mime type was accepted and added to the database. This test is a failure:

- An invalid mime type is in the database
- The user will not be informed of this

#### Requirement 2 - Remove Resource Type

**Test 1 input** Type = image/gif

**Expected Test output** The type is expected to be removed and the client returned to the file upload page

**Actual Test output** This test was a success

**Test 2 input** Type = wrong

**Expected Test output** Invalid mime type exception

**Actual Test output** The user is returned to the resources upload.  
This test is a failure:

- The user will not be informed of this

### Requirement 3 - Upload Resources

**Test 1 input** File = Desert.jpg Description = Test

**Expected Test output** The picture is expected to be uploaded and then shown.

**Actual Test output** This test was a success

**Test input** File = test.jpg Description = funny gif

**Expected Test output** An unsupported file exception should be thrown as the mime type image/gif is not in the database

**Actual Test output** An unsupported file exception was thrown. This test was a success

### Requirement 4 - Delete resource

**Test input** When viewed the uploaded resources have a delete button which will be tested

**Expected Test output** The resource is expected to be deleted and removed from view.



**Actual Test output** The resource is removed from the database and from view. This test was a success

### **15.2.2 Non-Functional Testing**

**Problem No editing of resource types** The only way to edit a mime type is to delete it then add it again with the new required size

**Example** There is no option to edit the size of an already established mime type

## **15.3 Resources Comparison**

## 16 Reporting Integration

The tests provided below show all cases in which the Reporting module was to make use of another Content module and the expected and actual results of each tests. We then explain which non-functional requirements have not been fulfilled in the overall Reporting module.

### 16.1 Reporting Team A

#### 16.1.1 Functional Testing

##### Requirement 1 - Get Thread Statistics

**Expected functionality** It provides statistical information that can be used in the interface to display facts about the average user and how the logged-in user compares with the average. Meant to provide a versatile way to get statistical information of subsets of posts complying with specified restrictions.

**Actual Test output** This test was a failure, due to the fact that the files that were meant to be uploaded for Group A reporting were not added to the final github commit, thus there are no files to test.

##### Requirement 2 - Upload CSV

**Expected functionality** It provides functionality to alter record sets in a Buzz space by uploading the relevant information that is stored in a csv file.

**Actual Test output** This test was a failure, due to the fact that the files that were meant to be uploaded for Group A reporting were not added to the final github commit, thus there are no files to test.

##### Requirement 3 - Get Thread Appraisal

**Expected functionality** Meant to provide a versatile way to get detailed or statistical information of subsets of posts complying with specified restrictions and their associated appraisals assigned by specified members.

**Actual Test output** This test was a failure, due to the fact that the files that were meant to be uploaded for Group A reporting were not added to the final github commit, thus there are no files to test.

#### **Requirement 4 - Export Thread Appraisal**

**Expected functionality** Meant to realise an off-line facility to apply a manual appraisal. It creates the dataset to be used that can be edited off-line and allow updates to be inserted through the importThreadAppraisal function.

**Actual Test output** This test was a failure, due to the fact that the files that were meant to be uploaded for Group A reporting were not added to the final github commit, thus there are no files to test.

#### **Requirement 5 - Import Thread Appraisal**

**Expected functionality** Meant to realise an offline-facility to apply a manual appraisal. It is dependent on the exportThreadAppraisal function.

**Actual Test output** This test was a failure, due to the fact that the files that were meant to be uploaded for Group A reporting were not added to the final github commit, thus there are no files to test.

#### **Requirement 6 - Export Thread**

**Expected functionality** Meant to provide means to back up the content of a thread or subset of a thread in a serialised text file.

**Actual Test output** This test was a failure, due to the fact that the files that were meant to be uploaded for Group A reporting were not added to the final github commit, thus there are no files to test.

#### **Requirement 7 - Import Thread**

**Expected functionality** Meant to provide means to restore the content of a thread or subset of a thread that was stored using the exportThread function.

**Actual Test output** This test was a failure, due to the fact that the files that were meant to be uploaded for Group A reporting were not added to the final github commit, thus there are no files to test.

**Test Coverage Analysis** This test was a failure, due to the fact that the files that were meant to be uploaded for Group A reporting were not added to the final github commit, thus there are no files to test. Thus there was a 0 percent completion rate for this phase of the mini project.

### **16.1.2 Non-Functional Testing**

**Problems** The files that were meant to be uploaded for Group A reporting were not added to the final github commit, thus there are no files to test.

**Performance** Due to this, there would be a problem with performance.

**Scalability** Due to this, there would be a problem with Scalability.

**Maintainability** Due to this, there would be a problem with Maintainability.

**Reliability** Due to this, there would be a problem with Reliability.

**Usability** Due to this, there would be a problem with Usability.

## **16.2 Reporting Team B**

### **16.2.1 Functional Testing**

If the use case implementation was successful:

#### **Requirement 1 - Get Thread Statistics**

**Expected functionality** It provides statistical information that can be used in the interface to display facts about the average user and how the logged-in user compares with the average.

**Actual Test output** This test was successful, however it provides minimal statistical information which is only limited to threads and does not include statistics related users.

### **Requirement 3 - Upload CSV**

**Expected functionality** It provides functionality to alter record sets in a Buzz space by uploading the relevant information that is stored in a csv file.

**Actual Test output** This test was successful. One is able to import that alters records and also export data.

### **Requirement 3 - Get Thread Appraisal**

**Expected functionality** Meant to provide a versatile way to get detailed or statistical information of subsets of posts complying with specified restrictions and their associated appraisals assigned by specified members.

**Actual Test output** This test was a failure, function was not implemented.

### **Requirement 4 - Export Thread Appraisal**

**Expected functionality** Meant to realise an off-line facility to apply a manual appraisal. It creates the dataset to be used that can be edited off-line and allow updates to be inserted through the importThreadAppraisal function.

**Actual Test output** This test was a failure, function was not implemented.

### **Requirement 5 - Import Thread Appraisal**

**Expected functionality** Meant to realise an offline-facility to apply a manual appraisal. It is dependent on the exportThreadAppraisal function.

**Actual Test output** This test was a failure, function was not implemented.

### **Requirement 6 - Export Thread**

**Expected functionality** Meant to provide means to back up the content of a thread or subset of a thread in a serialised text file.

**Actual Test output** This test was a failure, function was not implemented.

### **Requirement 7 - Import Thread**

**Expected functionality** Meant to provide means to restore the content of a thread or subset of a thread that was stored using the exportThread function.

**Actual Test output** This test was a failure, function was not implemented.

## **16.2.2 Non-Functional Testing**

**Problems** The reporting module was not integrated with the overall system. Due to this reason, the following were not able to be tested for.

**Performance** Module was not integrated and therefore there is no way of testing for performance.

**Scalability** Module was not integrated and therefore there is no way of testing for Scalability.

**Maintainability** Module was not integrated and therefore there is no way of testing for Maintainability.

**Reliability** Module was not integrated and therefore there is no way of testing for Reliability.

**Usability** Module was not integrated and therefore there is no way of testing for Usability.

### **16.3 Reporting Comparison**