Testing Report

Group Name: Testing Status

Daniel Christopher Alves Araüjo	13073878
Taariq Ghoord	10132806
Lerato Molokomme	11197961
Semaka Malapane	13081129
Mpedi Mello	11210754
Ryno Pierce	12003922
Lutfiyya Razak	10198408
Frederick Snyman	13028741
Keagan Thompson	13023782

Git repository link:
`https://github.com/u13073878/COS301-Testing-Status`

Date: 24 April 2015

Contents

1 assessProfile Use case **2**
1.1 group A - assessProfile 2
1.2 group B - assessProfile 2

2 setStatusCalculator Use case **3**
2.1 group A - setStatusCalculator 3
2.2 group B - setStatusCalculator 3

3 getStatusForProfile Use case **5**
3.1 group A - getStatusForProfile 5
3.2 group B - getStatusForProfile 5

4 createAppraisalType Use case **8**
4.1 group A - createAppraisalType 8
4.2 group B - createAppraisalType 8

5 activateAppraisalType Use case **10**
5.1 group A - activateAppraisalType 10
5.2 group B - activateAppraisalType 11

6 assignAppraisalToPost Use case **13**
6.1 group A - assignAppraisalToPost 13
6.2 group B - assignAppraisalToPost 13

1 assessProfile Use case

1.1 group A - assessProfile

Status A assesProfile function does not exists in the list of all the functions implemented. However the the function was used inside the function ThreadsDepthAssessor which made it difficult to test. ThreadsDepthAssesor was called and was used to test and the test failed.

1.2 group B - assessProfile

```
var test=require("unit.js");

exports.statusCalculator= function(test)
){

    test.expect(1);

    var expected=statusCalculatorRequest();
    var result=statusCalculatorResult();



    test.equal(expected.profileAssessor,result.profileAssessor);
    test.done();
)}
```

Figure 1: Unit tests for *setStatusCalculator*

2 setStatusCalculator Use case

2.1 group A - setStatusCalculator

2.2 group B - setStatusCalculator

Status B did not implement all the status calculators. Only ThreadDepth and numPost assessors are provided. The numPostAssessor counts the threads of a particular user and sets that as the status level of the user. The threadDepthAssessor calculates a users status level based on an average calculated by number of threads divided by a count of the total threads of a user. Unit tests for this will follow. It seems that the statusCalculatorRequest and statusCalculatorResult are implemented as functions so we cannot access their profileAssessors which must be the reason for the fail of the tests. Images are given below.

```
C:\Users\herb-adventure\Documents\COS301-Testing-Statu\StatusBTests>nodeunit setStatusCalculator.js

setStatusCalculator.js
? statusCalculator

Error: Expected 1 assertions, 0 ran
    at Object.test.done (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\types.js:121:25)
    at Object.exports.runTest (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:73:14)
    at C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:110:20
    at C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:513:13
    at iterate (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:123:13)
    at async.forEachSeries (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:139:9)
    at _concat (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:512:9)
    at Object.concatSeries (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:152:23)
    at Object.exports.runSuite (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:96:11)
    at Object.exports.runModule (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:158:13)
    at C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\nodeunit.js:75:21
    at C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:513:13
    at iterate (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:123:13)
    at async.forEachSeries (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:139:9)
    at _concat (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:512:9)
    at Object.concatSeries (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:152:23)
```

Figure 2: Unit tests for *setStatusCalculator test output*

```
TypeError: Cannot read property 'assess' of undefined
    at Object.exports.statusCalculator (C:\Users\herb-adventure\Documents\COS301-Testing-Statu\StatusBTests\setStatusCalculator.js:29:57)
    at Object.<anonymous> (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:236:16)
    at C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:236:16
    at Object.exports.runTest (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:70:9)
    at C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:118:25
    at C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:513:13
    at iterate (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:123:13)
    at async.forEachSeries (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:139:9)
    at _concat (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:512:9)
    at Object.concatSeries (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:152:23)
    at Object.exports.runSuite (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:96:11)
    at Object.exports.runModule (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:158:13)
    at C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\lib\nodeunit.js:75:21
    at C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:513:13
    at iterate (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:123:13)
    at async.forEachSeries (C:\Users\herb-adventure\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:139:9)


FAILURES: 2/2 assertions failed (313ms)
```

Figure 3: Unit tests for *setStatusCalculator test output*

3 getStatusForProfile Use case

3.1 group A - getStatusForProfile

3.2 group B - getStatusForProfile

Status B provided a *getStatusForProfile* as in Figure 41 of the master specifications. No pre- or post-conditions are tested for in the code written by Status B, however none were provided in the given master specifications.

The function that Status B provided takes as parameters the ID of the user being queried.

Nodeunit was used for doing unit tests on this code, and the code of the unit tests appears in the figure below:

```
var status = require('Status');

exports.getStatusForProfileTest1 = function(test){
    test.expect(1);
    status.getStatusForProfile("u00000006",function(res){
        test.equal(res,0);
        test.done()
    });
}

exports.getStatusForProfileTest2 = function(test){
    test.expect(1);
    status.getStatusForProfile("u00000001",function(res){
        test.equal(res,4);
        test.done();
    });
}

exports.getStatusForProfileTest3 = function(test){
    test.expect(1);
    status.getStatusForProfile("u00000002",function(res){
        test.equal(res,8);
        test.done();
    });
}

exports.getStatusForProfileTest4 = function(test){
    test.expect(1);
    status.getStatusForProfile("u00000003",function(res){
        test.equal(res,0);
        test.done();
    });
}

exports.getStatusForProfileTest5 = function(test){
    test.expect(1);
    status.getStatusForProfile("u00000004",function(res){
        test.equal(res,7);
        test.done();
    });
}

exports.getStatusForProfileTest6 = function(test){
    test.expect(1);
    status.getStatusForProfile("u00000005",function(res){
        test.equal(res,2);
        test.done();
    });
}
```

Figure 4: Unit tests for *getStatusForProfile*

Executing the tests above resulted in the following output:

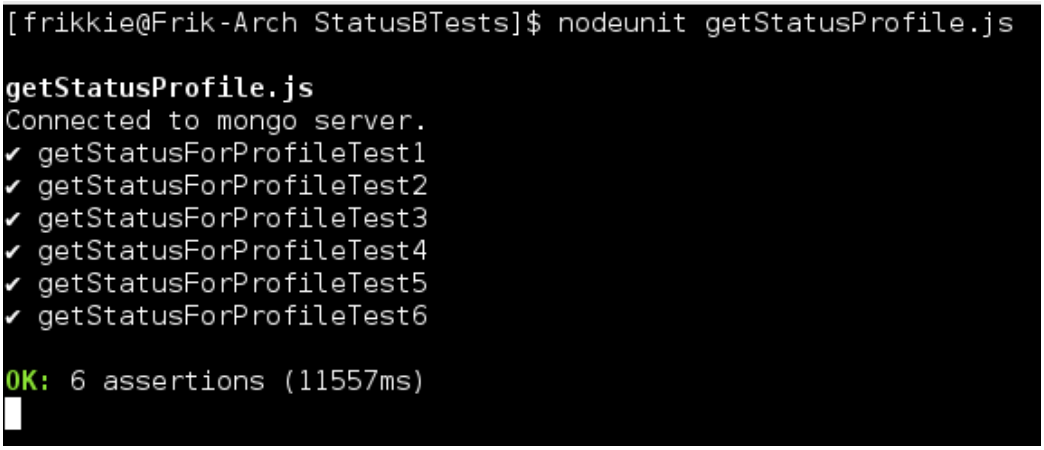

Figure 5: Results of the unit tests for *getStatusForProfile*

The tests that are run, checks the status value for six users present in the database, with differing status values. All six tests pass successfully and the function *getStatusForProfile* appears to hav been implemented correctly as per the master specification. These test cases cover 100% of the use cases as defined in the master specification.

4 createAppraisalType Use case

4.1 group A - createAppraisalType

4.2 group B - createAppraisalType

The *createAppraisalType* function created by Status B meets the requirements of the specification. The function implemented asks for the *name* and *description*. *activityPeriod* is also created. Therefore the function meets the requirements as indicated in figure 43 of the specification.

The *addAppraisalLevel* function created by Status B has a name and a level number. This meets the requirements.

The code for the unit tests is given below. The unit tests were made using Nodeunit.

```
/*
 test for createAppraisal
 */
exports.createAppraisalTest = function(test)
{
    test.expect(1);
    var expected = JSON.stringify({"name":"Funny","description":"humorous post"});
    var result = status.createAppraisal("Funny","humorous post");

    test.equal(result, expected);
    test.done();
}

/*
 test for createAppraisal
 */
exports.addAppraisalLevelTest = function(test)
{
    test.expect(1);
    var expected = JSON.stringify({rating: 1, rating_name: "Funny"});
    var result = status.addAppraisalLevel(1,"Funny");

    test.equal(result, expected);
    test.done();
}
```

Figure 6: Unit test for createAppraisalType and addAppraisalLevel

The following was the output of the createAppraisalType unit test created above:

```
C:\Users\Lesedi\WebstormProjects\statusTest>nodeunit createAppraisalType.js

createAppraisalType.js
? createAppraisalTest
? addAppraisalLevelTest

AssertionError: '[{"rating":1,"rating_name":"Funny"}]' == '{"rating":1,"rating_name":"Funny"}'
    at Object.equal (C:\Users\Lesedi\AppData\Roaming\npm\node_modules\nodeunit\lib\types.js:83:39)
    at Object.exports.addAppraisalLevelTest (C:\Users\Lesedi\WebstormProjects\statusTest\createAppraisalType.js:31:10)
    at Object.<anonymous> (C:\Users\Lesedi\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:236:16)
    at C:\Users\Lesedi\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:236:16
    at Object.exports.runTest (C:\Users\Lesedi\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:70:9)
    at C:\Users\Lesedi\AppData\Roaming\npm\node_modules\nodeunit\lib\core.js:118:25
    at C:\Users\Lesedi\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:513:13
    at iterate (C:\Users\Lesedi\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:123:13)
    at C:\Users\Lesedi\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:134:25
    at C:\Users\Lesedi\AppData\Roaming\npm\node_modules\nodeunit\deps\async.js:515:17


FAILURES: 1/2 assertions failed (30ms)
```

Figure 7: The result of the unit test

The *createAppraisalType* function meets the post condition that an appraisal is created (the function passes the unit test) when the function is called.

The *addAppraisalLevel* function doesn't meet the post condition, that is an appraisal level can't be created - the function fails the unit test.

5 activateAppraisalType Use case

5.1 group A - activateAppraisalType

Status A provides an *(*activateAppraisalType) function, but does not match the specification. The function implemented asks for the *(*appraisalType ID), but does not ask for a specific time period to be activated as indicated in figure 45 of the master specification. The code for the unit test is given by the figure below:

```
var settings = require('../buzz-settings/settings');
var database = require('../buzz-database/database')(settings);

var BuzzStatus = require('./buzzStatus')(database);



function testStatus()
{
    var tmp= BuzzStatus.activateAppraisalType({appraisalTypeID: "5513f7004af522f6583d9f14", spaceID: "COS 332"});

        console.log(tmp);
}

testStatus();
```

Figure 8: Unit test for activateAppraisalType

The following is the result of running the code:

```
connection error: [Error: failed to connect to [undefined:27017]]

C:\Users\Draak Koning\Desktop\UP\COS 301\cos301_Phase3\Handlebars Server\node_mo
dules\buzz-status>
```

Figure 9: Output of unit tests for activateAppraisalType

The first test, *activePeriodTest*, passes. *activePeriod()* takes two strings as parameters that represent the starting and ending date. However, it does no testing to ensure that the dates are in the correct format supported by JavaScript. Secondly, it does not have a callback parameter to specify a callback function, so it cannot be run asynchronously.

As you can see it could not connect to the database correctly in order to activate the appraisal type. Further inspection of the code show that it returns the value instead of using a callback function, this will cause it to always return empty or incorrect values as node runs asynchronously.

The implementation of the function does not meet the pre condition seeing that it doesn't test if the buzz space is active as well as, whether the active period is before the current date.

The post condition is net met due to the fact that the function could not connect to the database.

5.2 group B - activateAppraisalType

Status B did not provide an explicit *activateAppraisalType* function that matches Figure 45 of the project specifications. However, Status B does provide two functions, *activePeriod()* and *setAppraisal()*.

The code for the unit tests is given by the figure below. For the test, NodeUnit was used.

```
9
10  /*
11      test for activePeriod
12   */
13  exports.activePeriodTest = function(test)
14  {
15      test.expect(1);
16
17      var expectedResult = JSON.stringify({"active_from":"2015-10-01","active_to":"2015-10-30"});
18      var result = status.activePeriod("2015-10-01", "2015-10-30");
19
20      test.equal(result, expectedResult);
21      test.done();
22  }
23
24  /*
25      test for setAppraisal
26   */
27  exports.setAppraisalTest = function(test)
28  {
29      test.expect(1);
30
31      var post_id = "0";
32      var appraisalName = "Test";
33
34      status.setAppraisal(post_id, appraisalName); //The function does not return any value
35
36      test.done();
37  }
```

Figure 10: Unit test for activateAppraisalType

The following was the output given by NodeUnit.

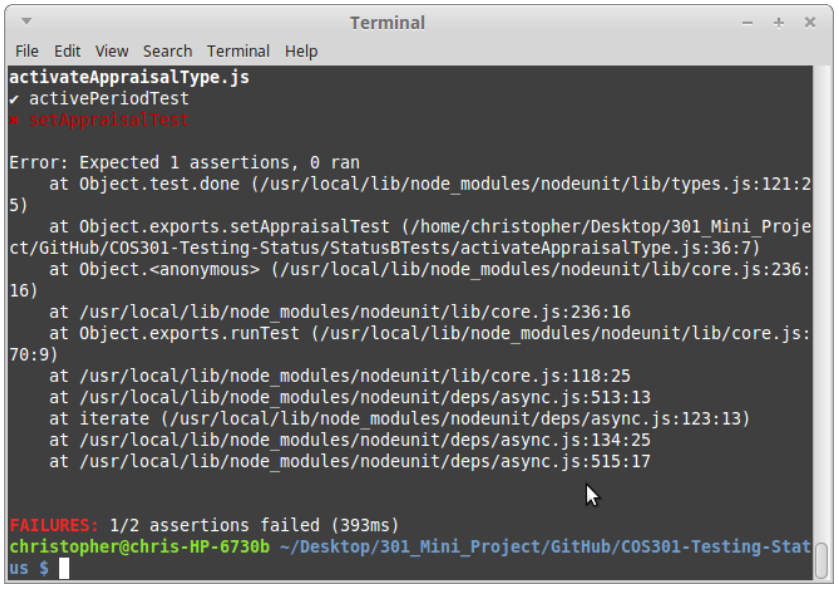

Figure 11: Output of unit tests for activateAppraisalType

The first test, *activePeriodTest*, passes. *activePeriod()* takes two strings as parameters that represent the starting and ending date. However, it does no testing to ensure that the dates are in the correct format supported by JavaScript. Secondly, it does not have a callback parameter to specify a callback function, so it cannot be run asynchronously.

The section function, *setAppraisal()*, fails. The function receives two parameters, a **post_id** and and **appraisal_name** to find a record in the database matching the **post_id** and assign the **appraisal_name** to **appraisal_id** in the database. The function has an error that attempts to assign a null value to **post.appraisal_id** in the database.

The pre-conditions are not tested in either function, although a assumption can be made about a buzz space being active. However, *setAppraisal()* does not test whether or not the active period is before the current date or time.

Due to the *setAppraisal()* failing, the function does not meet the post-condition of appraisalTypeAssignment being persisted to the database.

6 assignAppraisalToPost Use case

6.1 group A - assignAppraisalToPost

6.2 group B - assignAppraisalToPost

There is no explicit *assignAppraisalToPost* function available from Status B, however, they have provided a function *setAppraisal* which seems to attempt to fulfil that which is prescribed by *assignAppraisalToPost*, as seen on Figure 46 of the master specifications. The function *setAppraisal* receives two parameters, the first is the post ID which is used to identify the post, and the second is the Appraisal ID, which is used to identify the ID that is to be assigned.

Nodeunit was used for doing the unit tests.

Below is the code for the unit tests that was used for testing *setAppraisal*:

```
var status = require('Status');

exports.setAppraisalTest = function(test)
{
    test.expect(1);

    status.setAppraisal("post1", "Education");

    status.getPostAppraisal("post1",function(res){
        console.log(res);
        test.equal(res,"Education");
        test.done();
    });
}
```

Figure 12: Unit tests for *assignAppraisalToPost*

The figure below is the result of running the unit tests

```
[frikkie@Frik-Arch StatusBTests]$ nodeunit assignAppraisalToPost.js

assignAppraisalToPost.js
Connected to mongo server.
Education
✔ setAppraisalTest

OK: 1 assertions (7042ms)
```

Figure 13: Results of the unit tests for *assignAppraisalToPost*

The pre-conditions to be held as per the master specifications, is that a buzz space must be open before the function *assignAppraisalToPost* can be executed. It is not validated in the code that this pre-condition is upheld before continuing with execution. The post-condition hold, since the unit test passed.

The unit tests cover 100% of the use cases.