DocNo: 001.H.1.1

**Grape**

**Test Plan**

**Version 2.0**

**By**:

Group Undefined

2015-05

**Group Member**:

Hunter Lin

Birdy

Listen

Morning

Syachi

**Document Language**:

English

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 2015.5.24 | 1.0 | Initialization of the report | Hunter Lin |
| Final Date | 2.0 | Integrating all of the works | Hunter Lin |

**Key Word**

Grape, Defect

Black box testing, White box testing

Stub module, Driven module

Boundary testing

Unit test, Integration test, System test, Run time test, Stress test

**Abstract**

This document describes in detail the testing methodology and different test cases. It is of great important because of the high cost to pay if the hidden bug is found in the released version. So we need to take much attention on the designation and test cases of our software. The main contents include unit test, integration test, system test and stress test.

Content

[1. Introduction 6](#_Toc420232730)

[1.1. Purpose 6](#_Toc420232731)

[1.2. Background 6](#_Toc420232732)

[1.3. Definition 6](#_Toc420232733)

[1.4. Reference 6](#_Toc420232734)

[2. Test Plan 6](#_Toc420232735)

[2.1. Project Review 6](#_Toc420232736)

[2.2. Test Cases 6](#_Toc420232737)

[2.3. Unit Test 6](#_Toc420232738)

[2.3.1. Test Schedule 6](#_Toc420232739)

[2.3.2. Conditions 6](#_Toc420232740)

[2.3.3. Test References 6](#_Toc420232741)

[2.3.4. Test Training 6](#_Toc420232742)

[2.4. Integration Test 6](#_Toc420232743)

[2.4.1. Test Schedule 6](#_Toc420232744)

[2.4.2. Conditions 6](#_Toc420232745)

[2.4.3. Test References 6](#_Toc420232746)

[2.4.4. Test Training 7](#_Toc420232747)

[2.5. System Functional Test 7](#_Toc420232748)

[2.5.1. Test Schedule 7](#_Toc420232749)

[2.5.2. Conditions 7](#_Toc420232750)

[2.5.3. Test References 7](#_Toc420232751)

[2.5.4. Test Training 7](#_Toc420232752)

[2.6. Runtime Test 7](#_Toc420232753)

[2.6.1. Test Schedule 7](#_Toc420232754)

[2.6.2. Conditions 7](#_Toc420232755)

[2.6.3. Test References 7](#_Toc420232756)

[2.6.4. Test Training 7](#_Toc420232757)

[2.7. Stress Test 7](#_Toc420232758)

[2.7.1. Test Schedule 7](#_Toc420232759)

[2.7.2. Conditions 7](#_Toc420232760)

[2.7.3. Test References 7](#_Toc420232761)

[2.7.4. Test Training 7](#_Toc420232762)

[3. Test Design Specification 8](#_Toc420232763)

[3.1. Unit Test 8](#_Toc420232764)

[3.1.1. Control Method 8](#_Toc420232765)

[3.1.2. Test Case 8](#_Toc420232766)

[3.1.3. Process 8](#_Toc420232767)

[3.2. Integration Test 8](#_Toc420232768)

[3.2.1. Control Method 8](#_Toc420232769)

[3.2.2. Test Case 8](#_Toc420232770)

[3.2.3. Process 8](#_Toc420232771)

[3.3. System Functional Test 8](#_Toc420232772)

[3.3.1. Control Method 8](#_Toc420232773)

[3.3.2. Test Case 8](#_Toc420232774)

[3.3.3. Process 8](#_Toc420232775)

[3.4.Runtime Test 8](#_Toc420232776)

[3.4.1. Control Method 9](#_Toc420232777)

[3.4.2. Test Case 9](#_Toc420232778)

[3.4.3. Process 9](#_Toc420232779)

[3.5. Stress Test 9](#_Toc420232780)

[3.5.1. Control Method 9](#_Toc420232781)

[3.5.2. Test Case 9](#_Toc420232782)

[3.5.3. Process 9](#_Toc420232783)

[4. Criteria 9](#_Toc420232784)

[4.1. Scope 9](#_Toc420232785)

[4.1.1. Deflect verified rate criteria 9](#_Toc420232786)

[4.1.2. Coverage Rate Criteria 9](#_Toc420232787)

[4.2. Data Catalog 9](#_Toc420232788)

[4.3. Scale 9](#_Toc420232789)

[4.3.1. Test Ceasing Criteria 9](#_Toc420232790)

[4.3.2. Unit Test Ceasing Criteria 9](#_Toc420232791)

[4.3.3. Integration Test Ceasing Criteria 9](#_Toc420232792)

[4.3.4. System Test Ceasing Criteria 10](#_Toc420232793)

[5. Conclusion 10](#_Toc420232794)

Note:

黑色字部分为大家都需要写的。（大部分为功能测试的内容）

根据大家之前画use case时的分工，来写相应的system functional test.

黄色部分: Hunter Lin

蓝色部分: Morning

绿色部分: Birdy

红色部分: Listen

紫色部分: Syachi

**1. Introduction**

## 1.1. Purpose

This document is our test plan for the Grape System, which illustrates the details for the test context, test scope, test standard, and so on. This document will be the main reference for our testing. Therefore, the readers for this document are mainly the testers and the project manager of the Grape System.

Also we need to notice that, a small bug in the released version of the software will cost us much a lot than just several lines of code or apologizes.

## 1.2. Background

The system tested is named as “Grape”, which is developed by the Undefined Group (members are: Hunter Lin, Morning, Syachi, Listen, Birdy).

The Grape can be widely used in classroom and discussion room, the users can share their opinions and resources using this software as the communication platform. Also, the leader of the group can promulgate or share some important messages in the certain group.

The whole project began at April 5th. After requirement analysis, system designing, and coding, the next step is testing. The testing goes along in the computer center of Shanghai Jiaotong University. After coding out the system and our testers master the testing knowledge and skills, we can do our test.

## 1.3. Definition

Grape: A interactive software for resource sharing created by the Undefined group.

Defect: Software bug

Black box testing: A test method, which testers only pay attention to input and output.

White box testing: A test method, which testers must know the inside instruction of test object. Including branch testing, statement testing, path testing and so on.

Stub module: When taking unit testing and integration testing, the test object needs to call other unit, and then stub module can take instead of the called unit. It can be viewed as a **Proxy** pattern in the design pattern

Driven module: When taking unit testing and integration testing, the test object needs to make active by others, then driven module can take instead of the caller. It can also be viewed as a Proxy design pattern.

Test script: A small teat program for testing to call unit or be called by unit.

Equivalence partition: A test method in black box testing. It uses a set of values selected, instead of many input value, which are dealt with in the same way.

Boundary designing: It is the extension of the equivalence partition; usually it is the boundary of equivalent class.

Causation graph: When considering the relationship of each input, causation graph can show the combinations of all inputs and outputs.

Unit testing: Test on the smallest unit such as class in the software.

Integration testing: Test on the combination of several units to check if they can work together.

Regression testing: In integration testing, some integration test cases must be test again to check if they can work with other integrations.

System testing: Compared with requirement definition, look for some parts which are not coincident with the requirement.

Run time testing: Test if the request-response time reaches criteria.

Stress testing: Test if the system can afford heavy using stress.

WAS: Web Application Stress Tool, a testing tool for stress testing.

## 1.4. Reference

“Software Testing”

by Ron Patton

“Object-Oriented Software Engineering – Using UML, Patterns and Java”

by Allen H. Dutoit

**2. Test Plan**

## 2.1. Project Review

|  |  |  |
| --- | --- | --- |
| Function | Input | Output |
| Create group | groupName,topic,confirmMessage | a corresponding group in the database |
| Delete group | group\_id | a group deleted in the database |
| Search group | group\_id | the information of the group |
| Join group | group\_id | an association between the group and the current user is created in the database; |
| Quit group | group\_id | an association between the group and the current user is deleted in the database; |
| Create bulletin | user\_id, group\_id,  bulletin content | an association between the leader and the bulletin is created.  an association between the group and the bulletin is created.  Also note that, if the user is not the leader in the group, the creation should be denied. |
| Delete bulletin | user\_id, group\_id, bulletin\_id | Associations between the leader and the bulletin, group and bulletin should both be deleted. |

## 2.2. Test Cases

## 2.3. Unit Test

### 2.3.1. Test Schedule

### 2.3.2. Conditions

### 2.3.3. Test References

### 2.3.4. Test Training

## 2.4. Integration Test

All members in our team will participate in this test. Several units will be integrated as a part and be tested respectively.

### 2.4.1. Test Schedule

This test will be executed from 5.29 to 6.3.The work is to write stub module, driven module, test script, design test cases, and do the tests.

### 2.4.2. Conditions

1. 5 computers, 5 days to be used.
2. All members of our team will participate in this test. We must know how to do black box test.

### 2.4.3. Test References

1. Grape requirement document.
2. The whole code including (XXXX要填充).
3. Integration test environment and the causation graph are the same with the unit test.

### 2.4.4. Test Training

NULL

## 2.5. System Functional Test

### 2.5.1. Test Schedule

### 2.5.2. Conditions

### 2.5.3. Test References

### 2.5.4. Test Training

## 2.6. Runtime Test

### 2.6.1. Test Schedule

### 2.6.2. Conditions

### 2.6.3. Test References

### 2.6.4. Test Training

## 2.7. Stress Test

### 2.7.1. Test Schedule

### 2.7.2. Conditions

### 2.7.3. Test References

### 2.7.4. Test Training

**3. Test Design Specification**

## 3.1. Unit Test

### 3.1.1. Control Method

### 3.1.2. Test Case

// note: this part is the combination of 3.1.1(input) & 3.1.2(output) in the demo doc. I think it’s better to integrate them and create a table to illustrate. Like this:

|  |  |  |
| --- | --- | --- |
| Test case number | Input | Output |
| 1 |  |  |
| 2 |  |  |

### 3.1.3. Process

## 3.2. Integration Test

According to system business tier , present tier ,and subsystem , integrate related units to test the integration version. Use black box testing to check the function and action of integration version. The whole process employs bottom – top integration. Testers must write proper stub module, driven module, and test script.

### 3.2.1. Control Method

Every integration component is tested manually by testers. Since in our developing process the developersare just the testers, they can fix bugs right now once they find a bug.

### 3.2.2. Test Case

|  |  |  |
| --- | --- | --- |
| Test case number | Input | Output |
| 1 | Operations about group in the webpage | Corresponding respond in the front-end and the database |
| 2 | Operations about discussion in the webpage | Corresponding respond in the front-end and the database |
| 3 | Operations about vote in the webpage | Corresponding respond in the front-end and the database |
| 4 | Operations about user himself in the webpage | Corresponding respond in the front-end and the database |

### 3.2.3. Process

1. Design test cases.
2. Write stub module, driven module and test script. Create a database for test.
3. Run server, Execute code, and compare result with expected.
4. Fix bugs found, and continue testing till there are no bugs.
5. When no bug is found, the test is over.

## 3.3. System Functional Test

// note: there might be several cases in expansion. Add them by yourself.

### 3.3.1. Login

### 3.3.1.1. Control Method

### 3.3.1.2. Test Case

### 3.3.1.3. Process

### 3.3.2. Create a Bulletin

This part is tested by Hunter Lin in purpose of checking whether the bulletin can be created by the authorized user and displayed in the right way.

### 3.3.2.1. Control Method

First manually manage to create bulletins in different groups, where the user has different role in the certain group, check the result in both web browser and the database.

Then I wrote a program in Python, with MySQLdb to insert bulletins automatically, then examine the result in the database and web browser.

### 3.3.2.2. Test Case

|  |  |  |
| --- | --- | --- |
| Test Case Number | Input | Output |
| 1 | User\_id, Group\_id  (the user is the group leader) | A new item in bulletin table is inserted;  An association between the bulletin and the group is created. |
| 2 | User\_id, Group\_id  (the user is only a group member) | No change in databases.  Report “No authority” in the front end. |
| 3 | User\_id, Group\_id  (the user is not a member in the group.) | No change in databases.  Report “No authority” in the front end. |

### 3.3.2.3. Process

1. Design test cases.
2. Run test server and database server.
3. Manually input bug information, or use a program to input automatically. Execute test cases, and record bugs found.
4. After all the test cases reach system testing ceasing criteria, this test is over.

### 3.3.3. Delete a bulletin

This part is tested by Hunter Lin in purpose of checking whether the bulletin can be deleted successfully if the user is authorized to do it.

### 3.3.3.1. Control Method

First manually manage to delete bulletins in different groups, where the user has different role in the certain group, check the result in both web browser and the database.

Then I wrote a program in Python, with MySQLdb to delete bulletins automatically, then examine the result in the database and web browser.

### 3.3.3.2. Test Case

|  |  |  |
| --- | --- | --- |
| Test Case Number | Input | Output |
| 1 | User\_id, Group\_id, Bulletin\_id  (the user is the group leader and the bulletin is created by the leader himself) | The corresponding item is deleted from the bulletin table.  The corresponding association between the group and the bulletin is deleted. |
| 2 | User\_id, Group\_id, Bulletin\_id  (the user is the group leader but the bulletin is not created by the leader himself) | No change in databases.  Report “No authority” in the front end. |
| 3 | User\_id, Group\_id, Bulletin\_id  (the user is the group leader but the bulletin is not created by the leader himself) | No change in databases.  Report “No authority” in the front end. |
| 4 | User\_id, Group\_id, Bulletin\_id  (the user is the group leader but the bulletin is not created by the leader himself) | No change in databases.  Report “No authority” in the front end. |

### 3.3.3.3. Process

1. Design test cases.
2. Run test server and database server.
3. Manually input bug information, or use a program to input automatically. Execute test cases, and record bugs found.
4. After all the test cases reach system testing ceasing criteria, this test is over.

**3.4. Runtime Test**

### 3.4.1. Control Method

### 3.4.2. Test Case

### 3.4.3. Process

## 3.5. Stress Test

### 3.5.1. Control Method

### 3.5.2. Test Case

### 3.5.3. Process

**4. Criteria**

## 4.1. Scope

The coverage rate of test cases must reach 100%.

### 4.1.1. Deflect verified rate criteria

1-class and 2-class defect verified rate must reach 100%.

3-class and 4-class defect verified rate must reach 80%.

5-class defect verified rate must reach 60%.

### 4.1.2. Coverage Rate Criteria

Coverage rate of sentences must reach 80%.

Coverage rate of test cases must reach 100%.

Coverage rate of requirement testing must reach 100%.

## 4.2. Data Catalog

Testers use a testing program specifically designed for different test purposes. The test program will report success if the actual output is identical to the expected output. It reports error otherwise.

Finally, the success rate will be calculated and shown on the screen.

## 4.3. Scale

### 4.3.1. Test Ceasing Criteria

1. After unit testing, integration testing, system testing, test has reached unit testing ceasing criteria, integration testing ceasing criteria, and system testing ceasing criteria.
2. The software passes validation testing, and generates validation test conclusion.

### 4.3.2. Unit Test Ceasing Criteria

### 4.3.3. Integration Test Ceasing Criteria

1. Integration test cases have accessed.
2. According to integration test cases, testers have finished all the tests of integration.
3. Reach the coverage rate criteria of integration testing.
4. Make sure that more than 2 errors should be found every KLOC of integration versions.
5. Integration version function and capability must be consistent with definition.
6. All the defects have been verified, and the verified rate has reached the criteria.

### 4.3.4. System Test Ceasing Criteria

1. System test cases have accessed.
2. According to system test plan, testers have finished all the tests of the system.
3. Reach the coverage rate criteria of system testing.
4. Make sure that more than 1 error should be found every KLOC of the system.
5. System function and capability must be consistent with requirement documents.
6. All the defects have been verified, and the verified rate has reached the criteria.

**5. Conclusion**