DocNo: 001.I.1.1

**Grape**

**Test Analysis Report**

**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 |
| 2015.5.24 | 1.1 | Finish test unit on integration test and test cases predefined by Morning    sd the | Morning |
| 2015.5.24 | 1.2 | Finish the document on unit test and the test cases predefined by Birdy | Birdy |
| 2015.5.24 | 1.3 | Finish the document on stress test and the test cases predefined by Listen | Listen |
| 2015.5.24 | 1.4 | Finish the document on system functional test and runtime test and the test cases predefined by Syachi | Syachi Cui |
| 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](#_Toc422338056)

[1.1. Purpose 6](#_Toc422338057)

[1.2. Background 6](#_Toc422338058)

[1.3. Definition 6](#_Toc422338059)

[1.4. Reference 7](#_Toc422338060)

[2. Test Overview 7](#_Toc422338061)

[3. Test Result & Findings 7](#_Toc422338062)

[3.1. Unit Test 7](#_Toc422338063)

[3.1.1. Test Case 7](#_Toc422338064)

[3.1.2. Process 8](#_Toc422338065)

[3.1.3. Compare with Test Plan 9](#_Toc422338066)

[3.2. Integration Test 9](#_Toc422338067)

[3.2.1. Test Case 9](#_Toc422338068)

[3.2.2. Process 9](#_Toc422338069)

[3.2.3. Compare with Test Plan 9](#_Toc422338070)

[3.3. System Functional Test 9](#_Toc422338071)

[3.4. Runtime Test 9](#_Toc422338072)

[3.4.1. Test Case 10](#_Toc422338073)

[3.4.2. Process 10](#_Toc422338074)

[3.4.3. Compare with Test Plan 10](#_Toc422338075)

[3.5. Stress Test 10](#_Toc422338076)

[3.5.1. Test Case 10](#_Toc422338077)

[3.5.2. Process 10](#_Toc422338078)

[3.5.3. Compare with Test Plan 11](#_Toc422338079)

[4. Function Test Report 11](#_Toc422338080)

[4.1. Login 11](#_Toc422338081)

[4.1.1. Test Case 11](#_Toc422338082)

[4.1.2. Limitation 11](#_Toc422338083)

[4.2. Group Operation 11](#_Toc422338084)

[4.2.1. Test Case 11](#_Toc422338085)

[4.2.2. Limitation 12](#_Toc422338086)

[4.3. Bulletin Operation 12](#_Toc422338087)

[4.3.1. Test Case 12](#_Toc422338088)

[4.3.2. Limitation 13](#_Toc422338089)

[4.4. Vote Operation 13](#_Toc422338090)

[4.4.1. Test Case 13](#_Toc422338091)

[4.4.2. Limitation 14](#_Toc422338092)

[5. Analysis Abstracts 14](#_Toc422338093)

[5.1. Capacity 14](#_Toc422338094)

[5.2. Flaws & Limitations 14](#_Toc422338095)

[5.2.1. Some Existing Problems 14](#_Toc422338096)

[5.2.2. Some Unrealized Functions 14](#_Toc422338097)

[5.3. Suggestions 14](#_Toc422338098)

[5.4. Evaluation 14](#_Toc422338099)

[6. Test Cost 14](#_Toc422338100)

Task Distribution:

Hunter: Orange

Syachi: Purple

Morning: Blue

Birdy: Green

Listen: Red

A **BIGGGG** Note:

加粗部分的内容由大家一起完成，具体分工如下:

1. Function Test:

每个人写自己所测试的那一部分。如果没有这部分内容，按照格式自己写（其实就是之前那份报告里面，还有没完成的部分。）

2. Analysis Abstracts – Capacity:

这一部分每个人将自己所实现的功能概括性地写上去即可。（当然多扯一点总是好的）

**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 Overview**

**3. Test Result & Findings**

## 3.1. Unit Test

### 3.1.1. Test Case

|  |  |  |
| --- | --- | --- |
| Test case number | Input | Output |
| 1 | Sign up with proper mail, username and password. | Sign up successfully. |
| 2 | Sign up with wrong mail address. | Can’t sign up and remind that the mail address is wrong. |
| 3 | Sign up with wrong username. | Can’t sign up and remind that the username has been used. |
| 4 | Sign up with different password and confirming password. | Can’t sign up and remind that the password and confirming password are different. |
| 5 | Log in with proper username and pass word. | Log in successfully. |
| 6 | Log in with wrong username. | Can’t log in and remind that the username is wrong. |
| 7 | Log in with wrong password. | Can’t log in and remind that the password is wrong |
| 8 | Create new group with proper group name, topic, description and confirm message. | Create a new group successfully and other users can have access to the information of the group. The creator is appointed as the leader. |
| 9 | Search a group with the group id. | If the group id exists, you will find the group information. Otherwise, you will get nothing. |
| 10 | Attend the group with confirmed message. | If the confirm message is right, you will attend the group successfully. Otherwise, you will fail to attend. |
| 11 | Leader creates a vote. | Voting will be published onto the Voting Board. The members in the group have access to the vote. |
| 12 | Members attend the vote. | The system will receive the members’ votes and make a statistic after the voting ends. |
| 13 | Member in the group generates a question. | Question will be published onto the Discussion Board. The members in the group have the access to the question and can reply to it. |
| 14 | Member in the group replies to the question. | The reply will be published onto the Question sub interface in Discussion Board. The questioner will receive message and members in the group have access to the reply. |
| 15 | Admin delete user or group. | The user account and group id will be invalid. |

### 3.1.2. Process

1. Design test cases.
2. Write stub module, driven module and test script.
3. 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.

### 3.1.3. Compare with Test Plan

## 3.2. Integration Test

### 3.2.1. 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 |

### 3.2.2. 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.

### 3.2.3. Compare with Test Plan

## 3.3. System Functional Test

Please refer to part 4 (Function Test Report) for details.

## 3.4. Runtime Test

### 3.4.1. Test Case

Every model must choose the test case whose run time is longest, and if this is less than 20 second, the model passes the criteria.

### 3.4.2. Process

1. Set up a complete database.
2. Design test cases.
3. Have tests, and record bugs.
4. After the test reaches run time criteria, test is over.

### 3.4.3. Compare with Test Plan

## 3.5. Stress Test

Use LOCUST which is deemed to be awesome by the author of Flask, Jinja2 to test.

### 3.5.1. Test Case

|  |  |  |
| --- | --- | --- |
| Test case number | Input | Output |
| 1 | Simulate this case that at a time 1000 users visit the server, and distribute the flux in different page groups. | LOCUST provide us with the ReponseContextManager class to see the request result and whether it was successful. |
| 2 | Throttle bandwidth to test the capability when user takes dial-up or other connection to surf on Internet. |

### 3.5.2. Process

a) Set up a complete database

b) Design test cases

c) Run server

d) Have tests by LOCUST, and record defects

e) After the test reaches stress criteria, test is over.

### 3.5.3. Compare with Test Plan

**4. Function Test Report**

## 4.1. Login

### 4.1.1. Test Case

|  |  |  |
| --- | --- | --- |
| Test Case | Input | Output |
| Login | wrong user ID | Show that your user id is wrong. |
| Login | wrong password | Show that your pw is wrong |
| Login | NULL user ID | Show that user ID cannot be NULL |
| Login | NULL password | Show that password cannot be NULL |
| Login | correct user ID and pw | Show user index |
| Login | correct admin ID and pw | Show admin index |

### 4.1.2. Limitation

## 4.2. Group Operation

### 4.2.1. Test Case

|  |  |  |
| --- | --- | --- |
| Function | Input | Output |
| Create group | groupName,topic,confirmMessage | a corresponding group in the database |
| Create group | groupName,topic,confirmMessage same as previous one | report that the group already exists |
| delete group | correct group\_id | a group deleted in the database |
| delete group | wrong group\_id | report fail to delete group due to authority or other errors |
| search group | correct group\_id | the information of the group |
| search group | wrong group\_id | return no information found |
| join group | correct group\_id | an association between the group and the current user is created in the database; |
| join group | wrong group\_id | report fail to join group |
| quit group | group\_id | an association between the group and the current user is deleted in the database; |

### 4.2.2. Limitation

## 4.3. Bulletin Operation

### 4.3.1. Test Case

|  |  |  |
| --- | --- | --- |
| Test Case Number | Input | Output |
| Create a bulletin | 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. |
| Create a bulletin | User\_id, Group\_id  (the user is only a group member) | No change in databases.  Report “No authority” in the front end. |
| Create a bulletin | User\_id, Group\_id  (the user is not a member in the group.) | No change in databases.  Report “No authority” in the front end. |
| Delete a bulletin | 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. |
| Delete a bulletin | 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. |
| Delete a bulletin | 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. |
| Delete a bulletin | 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. |

### 4.3.2. Limitation

## 4.4. Vote Operation

### 4.4.1. Test Case

|  |  |  |
| --- | --- | --- |
| Test Case Number | Input | Output |
| create vote | vote\_content,vote\_options,vote\_timelimit | A corresponding vote in the database |
| create vote | timelimit not set or empty vote\_options and vote\_content | Ban user to submit |
| delete vote | vote\_id | Delete all the corresponding information in the database. |
| delete vote | wrong vote\_id | report fail to delete group due to authority or other errors |
| operate vote | option,vote\_id | Update the votes of the corresponding option by add 1 in the database and insert the record what the option the user votes. |
| operate vote | option,wrongvote\_id | report fail to vote due to authority or other errors |
| View vote voted | vote\_id | Show the vote and the option voted |
| view vote voted | wrong vote\_id | Report fail to view the option voted due to authority or other errors |
| display vote result | vote\_id | One bar graph displaying the distribution of the votes over different options in the database. |
| finish vote | vote\_timelimit | The database automatically set the vote status to 0 which means the end by using the event of MySQL. |

### 4.4.2. Limitation

// Create something new by yourself.

**5. Analysis Abstracts**

## 5.1. Capacity

## 5.2. Flaws & Limitations

### 5.2.1. Some Existing Problems

### 5.2.2. Some Unrealized Functions

## 5.3. Suggestions

## 5.4. Evaluation

**6. Test Cost**