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
2015.5.24	1.1	Finish test unit on integration test and test cases predefined by Morning	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

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.
create vote	vote_content,vote_options, vote_timelimit	A corresponding vote in the database
delete vote	vote_id	Delete all the corresponding information in the database.
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.
View vote voted	vote_id	Show the vote and the option voted
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.
Create a discussion	group_id, user_id, discussion content	a corresponding discussion in the database
Reply to a discussion	group_id,user_id,discussion_id , reply content	a corresponding reply in discussion part in the database
Delete a discussion	group_id,user_id,discussion_id	a discussion deleted in the database

2.2. Test Cases

Test Name	Test Procedure	Test Context & Purpose
Unit testing	6.9~6.15	A set of both white box testing and black box testing to check every use case whether it can run successfully and as we expected. This test goes with coding.
Integration testing	6.16~6.20	A set of black box testing to test the unit interface and integration function,

		and run regression test on previous integration.
System Functional testing	6.21~6.23	A set of black box testing to demonstrate conformance with requirement. To check that: all functional requirements satisfied, all performance requirement achieved. If permitted, alpha testing can be used.
Run time testing	6.23~6.26	Make use of black box testing to check run time of this software whether to satisfy the criteria of requirement analysis.
Stress testing	6.27~7.1	Test the capability of server and other extreme conditions that will be met when finally put into use. For example, the capability when many users visit server, the maximum storage spaces available.

2.3. Unit Test

All members of team Grape will participate in this test. Because of functional test, almost all parts of the software will be tested.

2.3.1. Test Schedule

This test dates from 6.9 to 6.15.The work is to write stub module, driven module, test script, design test cases, and have tests.

2.3.2. Conditions

a) 5 computers,5 days to be used.

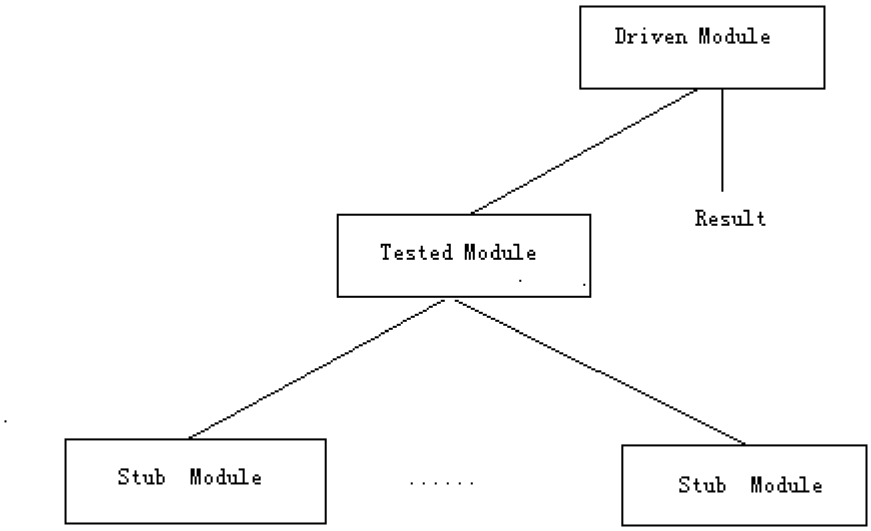
b) All members of the twister team will participate in this test. We must know how to do black box testing and white box testing

2.3.3. Test References

a) Grape requirement document

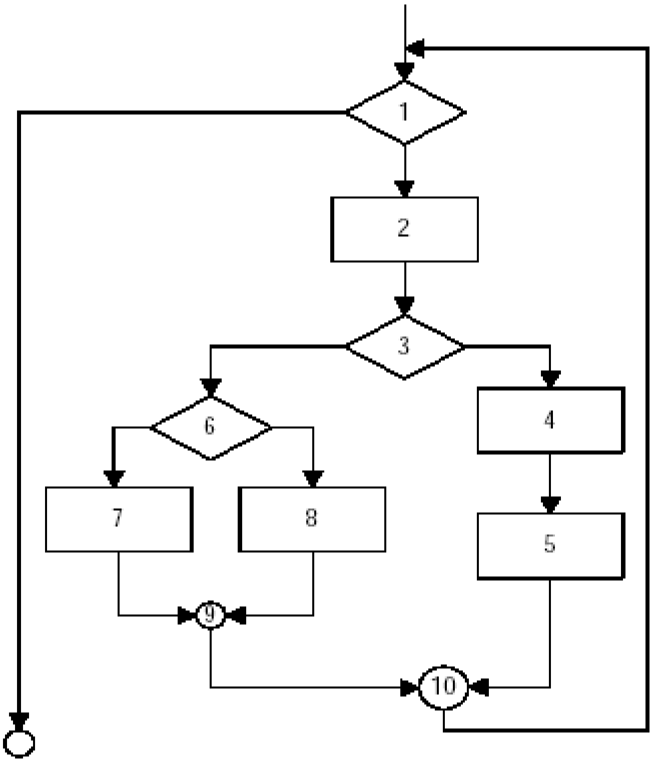
b) All program units such as front controller, group management delegate, voting operations, discussion operations and so on.

c) Unit test environment



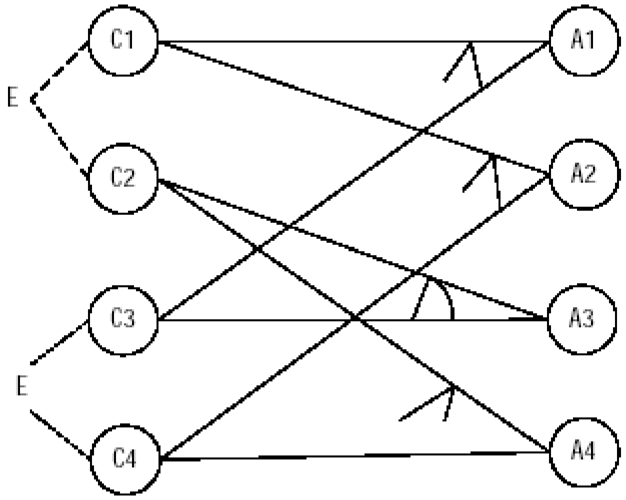
Unit test environment

d) When using white box testing, testers must translate program procedure into workflow graph:



Work Flow Graph

e) When testing on query module and statistics module , causation graph can be used:



Causation Graph

2.3.4. Test Training

NULL

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 6.16 to 6.20. The work is to write stub module, driven module, test script, design test cases, and do the tests.

2.4.2. Conditions

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

2.4.3. Test References

a) Grape requirement document.
b) The whole code including user operation, group operation, vote operation, discuss operation and message operation.

c) 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

All members of the team will participate in this test. The whole software will be tested.

2.5.1. Test Schedule

This test dates from 6.21 to 6.23.The work is to design test cases, and take testing.

2.5.2. Conditions

a) 2 computers(one as server and another as client).
b) All members of the team will participate in this test. We must know how to do black box test.

2.5.3. Test References

a) Grape requirement document.
b) Whole Grape code including (Login, Register, Log out, Search for a group, Attend a group, Quit a group, Raise a vote, Vote for a vote, View voting results, Raise a discussion, Reply a discussion, Delete a discussion).

2.5.4. Test Training

NULL

2.6. Runtime Test

All members of the twister team will participate in this test. All parts of the software will be tested.

2.6.1. Test Schedule

This test dates from 6.24 to 6.26.The work is to design test cases, and take testing.

2.6.2. Conditions

a) 2 computers(one as server and another as client).
b) All members of the team will participate in this test. We must know how to do black box test.

2.6.3. Test References

a) Grape requirement document.
b) Whole Grape code including (Login, Register, Log out, Search for a group, Attend a group, Quit a group, Raise a vote, Vote for a vote, View voting results, Raise a discussion, Reply a discussion, Delete a discussion).

2.6.4. Test Training

NULL

2.7. Stress Test

All members of the grape team would participate in the test. The test aims to the load capacity of the server.

2.7.1. Test Schedule

This test dates from 6.27 to 7.1. The work is to use LOCUST to simulate lots of simultaneous users to achieve the goal of stress test.

2.7.2. Conditions

a) All members would participate in and install the LOCUST loading testing tool and PYZMQ to run LOCUST distributed.
b) 2 computers (CPU: P4, Memory: 512M),5 days to be used.

2.7.3. Test References

Grape requirement document.

2.7.4. Test Training

NULL

3. Test Design Specification

3.1. Unit Test

The smallest test unit is class. Use white box testing to test inside instruction of unit, and black box testing to test function and action of test object. Unit testing goes with developing. Testers must write proper stub module, driven module, and test script.

3.1.1. Control Method

Each unit is tested manually by tester. Because in our developing process the developer is just the tester, they can fix bugs right now once they find a bug.

3.1.2. 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.

3.1.3. Process

a) Design test cases.
b) Write stub module, driven module and test script.
c) Execute code, and compare result with expected.
d) Fix bugs found, and continue testing till there are no bugs.
e) When no bug is found, the test is over.

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 developers are 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

a) Design test cases.
b) Write stub module, driven module and test script. Create a database for test.
c) Run server, Execute code, and compare result with expected.
d) Fix bugs found, and continue testing till there are no bugs.
e) When no bug is found, the test is over.

3.3. System Functional Test

3.3.1. Login

This part is tested by Syachi Cui in purpose of checking whether the user can be created via the register page and login in the right way.

3.3.1.1. Control Method

First manage to create users using MySQL client. Check the result in both web browser and the database.

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

3.3.1.2. 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

3.3.1.3. Process

a) Set admin ID and password, insert into the staff table some staff info.
b) Design test cases.
c) Run test server and database server.
d) Manually input user ID and password, execute test cases, and record bugs found.
e) After all the test cases reach system testing ceasing criteria, this test is over.

3.3.2. Group Operation

This part is tested by morning. The goal is to test the functions concerning group including join, quit, create, delete and so on.

3.3.2.1. Control Method

Manual operations on the web page. The testing result will be recorded in Excel.

3.3.2.2. 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;

3.3.2.3. Process

a) Generate some pre-defined information about users and groups in the database.
b) Design test cases.
c) Run web server and database server.
d) Manually execute the operations about group at the front-end, and record bugs found.
e) After all the test cases reach system testing ceasing criteria, this test is over.

3.3.3. Bulletin Operation

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.

3.3.3.1. 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.

3.3.3.2. 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	The corresponding item is deleted from the bulletin table.

	and the bulletin is created by the leader himself)	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.

3.3.3.3. Process

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

3.4.3. Vote Operation

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

3.3.4.1. 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.

3.3.4.2. 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.

3.3.4.3. Process

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

3.4. Runtime Test

This part is tested going with System test, to check if system run time reaches the run time criteria, that is: The system response time (response to users' request from servers to clients) must less than 20 second, if the network state is normal.

3.4.1. Control Method

Manually have tests, and the testing result will be recorded in Excel.

3.4.2. 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.

3.4.3. Process

a) Set up a complete database.
b) Design test cases.
c) Have tests, and record bugs.
d) After the test reaches run time criteria, test is over.

3.5. Stress Test

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

3.5.1. Control Method

Use LOCUST to run stress test automatically and generate test result by LOCUST.

3.5.2. 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.	

3.5.3. 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.

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

a) After unit testing, integration testing, system testing, test has reached unit testing ceasing criteria, integration testing ceasing criteria, and system testing ceasing criteria.
b) The software passes validation testing, and generates validation test conclusion.

4.3.2. Unit Test Ceasing Criteria

a) Unit test cases have accessed.
b) According to unit test cases, testers have finished all the tests of units.
c) Reach the coverage rate criteria of unit testing.
d) Make sure that more than 3 errors should be found every KLOC of units.
e) Unit function must be consistent with design model.
f) All the defects have been verified, and the verified rate has reached the criteria.

4.3.3. Integration Test Ceasing Criteria

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

4.3.4. System Test Ceasing Criteria

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