**CS673S16 Software Engineering**

**Team X - Project Name**

**Tests Report**

****

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Xiang Chen | Configuration Leader;  Integration Environment Leader | *Xiang Chen* | 10/24/2017 |
| Yansen Liu | Design leader | *Yansen Liu* | 10/26/2017 |
|  |  |  |  |
| Weicheng Yu | Backup Team Leader; Security Leader; | *Weicheng Yu* | 10/26/2017 |
| Lu Min | QA Leader | *Lu Min* | 10/26/2017 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| **alpha** | **Team 3** | **10/30/2017** | **Complete main features** |
|  |  |  |  |

[Introduction](#_87t9hln2vjz0)

[Test Summary](#_sm5odwyvuk3j)

[Tests Reports](#_2iy9xpvb9o9s)

[Testing Metrics](#_mtfbusfb0eq3)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Introduction (Xiang)

In this section, give a summary of this tests report document.

In order to detect fault/error/defects and provoke failures in a planned way, we followed the test driven development for our project. We conduct unit test, module test, integration test, system test and acceptance test on our project and try to solve the bugs / defects that comes from this process. This report is aim to describe the whole test process for different components with details and explanation.

# Test Summary (Xiang)

In this section, you will summarize what was tested and what happened, based on each test type.

* + Unit Tests

**Test Scope**: Individual component (class or subsystem)

**Purpose**: Confirm that the component or subsystem is correctly coded and carries out the intended functionality

**Example and Practice**:

1. Each php action function like create\_story(), delete\_story(), sort\_by\_hand()
2. Each php condition function like isMobile(), isPassDue(), hasTag()
3. Each javascript function like pop\_windows(), regular\_check()
4. Each JQuery function like hide(), show(), toggle()

**Result**: most of unit test has been passed successfully, few unit test has also been passed after some code change and modification.

* + System Tests

**Test Scope**: The entire system (on test bed)

**Purpose**: Determine if the system meets the requirements (functional and nonfunctional)

**Example and Practice**:

1. Create single story , edit it and delete it using dummy data
2. Sort different story based on their due day
3. Create single sprint , edit it and delete it using dummy data
4. Merge all story together and sort them by hand

**Result**:In some cases, the input with special characters can not be accepted. Then we modified the regular expression and also add pre-check function to the system. After that, the problem has been solved. All system test passed successfully.

* + Acceptance Tests

**Test Scope**: The entire system delivered (on final platform)

**Purpose**: Demonstrate that the system meets the requirements and is ready to use

**Example and Practice**:

1. Given a Project created and its subsidiary user story created, when I want to change specific user story description and input the new requirements or replace current description to an updated one into input box, then a clickable edit button displays , allowing me to change user story description. After clicking edit button, I can add, replace, delete descriptions.
2. Given a new user story was created, when a user added a description to the user story and clicked save button, then this user story would successfully have detailed description.
3. Given a project with some finished requirement, when the product owner want to delete some finished requirement and click on the delete button right on each finished requirement history, then the requirement history will be deleted and cannot restore.

**Result**: We passed most of the acceptance tests and meet their requirements. However, some features in the acceptance tests has not been implemented yet. Few features needs to be improved in the future. We plan to make some code change in the final iteration to pass all the acceptance tests.

# Tests Reports (Lu, Yuhao(not correct), Dawei, Yansen-> two test cases each person)

Example: <https://www.guru99.com/test-case.html>

(check the video and tables )

In this section, you will give a detailed description of each test case performed and the result. You shall list what are existing tests developed in the previous semester and what are new tests developed currently.

For each test case, you can use the following template (or something based on the following template)

* Test case ID, name
* New or old:
* Test items: (what do you test )
* Test priority (high/medium/low)
* Dependencies (to other test case/requirement if any):
* Preconditions: (if any)
* input data:
* Test steps:
* Postconditions:
* Expected output:
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes:

(You can use an additional table or document for this section)

All the test cases with details are listed in the link below using the template:

[Additional table link](https://docs.google.com/spreadsheets/d/1CGYv17k1YQjrvvOj_J151QLHhNvMrDJkOoC_8Az-tYg/edit#gid=0)

*(and we also record the error we will explain in the testing metrics part, including user data entry, code error and design. The environment part is not listed here, it is in the compatibility testing.)*

**Test case mind map:**

This following mind map leads us with orderly and concrete test cases. And it may also help others have an integral view of all the test cases.

The blue area represents iteration 1.

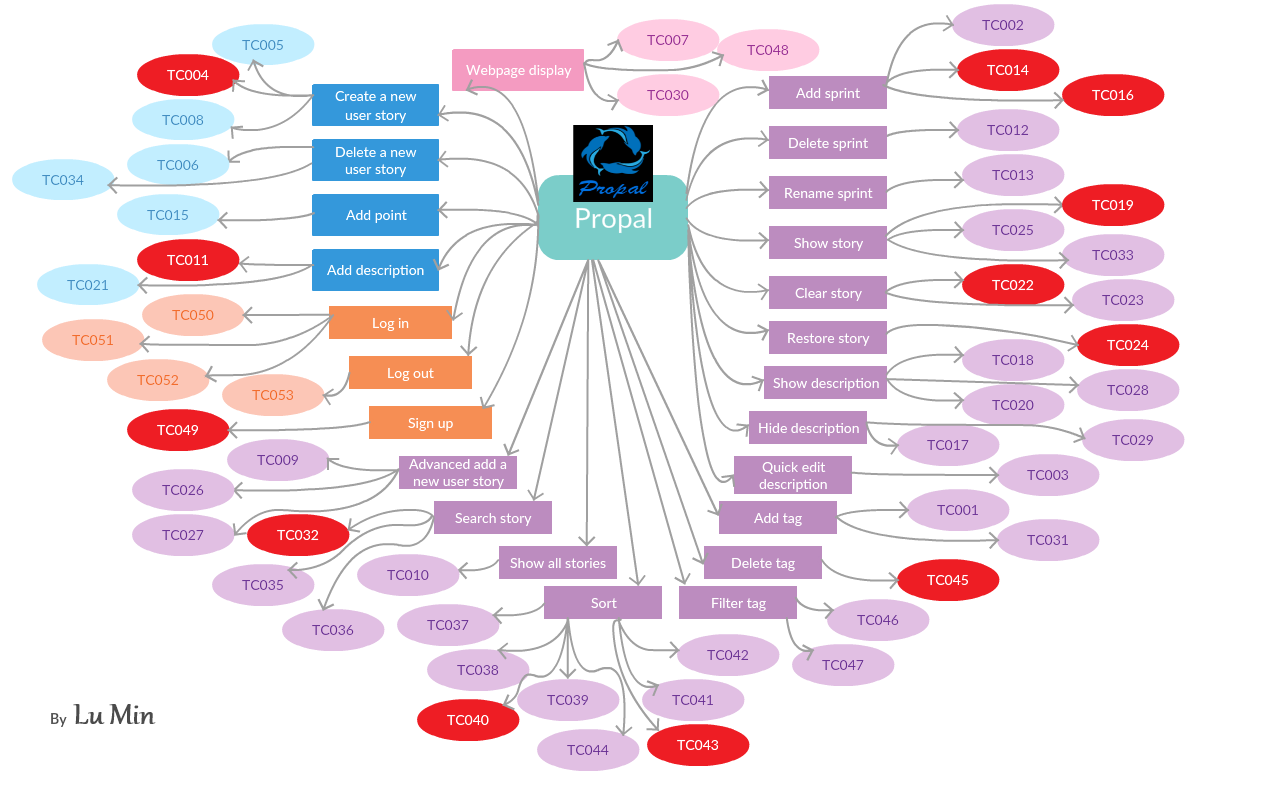
The purple area represents iteration 2.

The orange area represents iteration 3.

The pink area represents the basic display of our web application.

We extend from our project propal, all the rectangles mean all the user stories we implemented in our whole project. And the eclipses mean the test case code pointing to which user story. All the red eclipses mean these test cases failed.

And with the specific test case code, we can find the details in the above table link.



# Testing Metrics (Lu, Weicheng,Yuhao(missing); at least two table, two graph, with explanation)

// Instructions from professor

Table Example: <https://www.slideshare.net/deepikashanti/13-software-metrics>

(especially from page 16 to end)

Graph Example: <https://www.qasymphony.com/blog/64-test-metrics/>

In this section, you shall report the any metrics used for the evaluation, e.g. coverage, defects rate, etc.

// Actual stuff

**Testing tables**

Table 1. Size Oriented Metrics

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint # | Line of Code (lines) | Total Effort (man hour) | Document Produced  (pages) | Errors Found in Test (count) | Defects found in sprint  (count) | User Story Implemented  (count) |
| Sprint0 | 10 | 159 | 24 | 0 | 0 | 0 |
| Sprint1 | 1458 | 166 | 72 | 10 | 3 | 3 |
| Sprint2 | 3546 | 192 | 95 | 12 | 9 | 19 |
| Sprint3 |  |  |  |  |  |  |

Details on Table 1:

In this section, details on how each column is calculated will be explained.

Line of Code: Total lines of code on github before the first presentation. Sprint 0 also has some lines on github because each teammate try to commit on readme file.

Total Effort in Man Hour: Total actual hours on weekly report excel file for each sprint.

Document Produced: Total pages of documentation on google drive for each sprint.

Errors found in test cycle: Known bugs in each sprint that do not have time to resolve.

Defects found in sprint: Issues on github each sprint.

User Story Implemented: User Stories on pivotaltracker that were implemented in each sprint.

Table 2. Function Oriented Metrics

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Measurement Parameter | count |  | Weighing Factor | | |  | Subtotal |
| Simple | Average | Complex |  |  |
| No of User Story |  | x | 3 | 5 | 7 | = |  |
| No of Document |  | x | 1 | 3 | 5 | = |  |
| No of User |  | x | 4 | 6 | 8 | = |  |
| No of Day server is up |  | x | 1 | 3 | 5 | = |  |
| Count Total |  |  |  |  |  |  |  |

Details on Table 2:

In this section, details on how point is assigned will be explained.

This metrics will be used to calculate a general point indicating how functional our web application is when we publish it. Each measurement parameter will have a weighing factor that will be explained in details in below.

No of User Story: In this measurement parameter, we assign 3, 5, 7 points according to weighting factor to each no of user story we implement. Notice this will construct the second largest portion of our total score because we believe more user story means more feature and more feature means a better functional web application. Simple means this user story is part of another more general user story and it is relatively simple to implement without changing too much to structure and other part of the application.

No of Document:

Table 3: Domain testing boundary table

The following table is the domain testing boundary table, the variables in the table are important ones in the project. Because some variables don’t have specific boundaries, “\” means that there are no boundaries neither special cases.

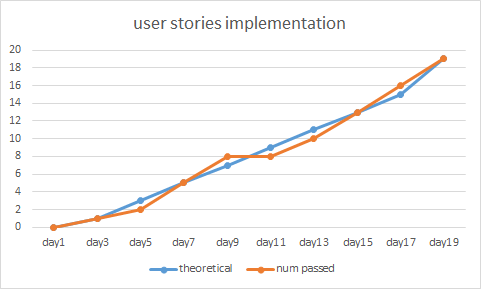
|  |  |  |  |
| --- | --- | --- | --- |
| Variables | Valid case equivalence classes | Invalid case equivalence classes | Boundaries and special cases |
| Username | Signed up users | Not signed up users | Signed up users |
| User story title | “test” | “ ” | \ |
| Sprint name | “Sprint 1” | “”, or too long | \ |
| User story tag | “core” | “”, or too long | \ |
| User story description | “Mainly for testing” | \ | \ |
| User story point | “+1”, “not selected” | “+6” | “+1”~”+3”, “not selected” |
| Search user story | “test” | “ ”, or user story not exist | Completed User stories |
| Shown tag | “Core”, “minor” | Not created tag | Created tags |

**Testing graphs**

Graph 1:

The following graph is the user stories implementation during out project, and the blue line is the theoretical line of the number of user stories we expected, and the orange line is the actual number of user stories we passed during project time.

Iteration 2:



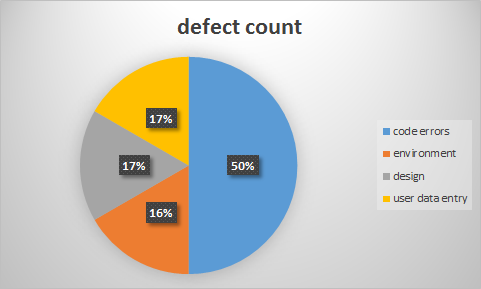
Graph 2:

We roughly divide the defect into four types.

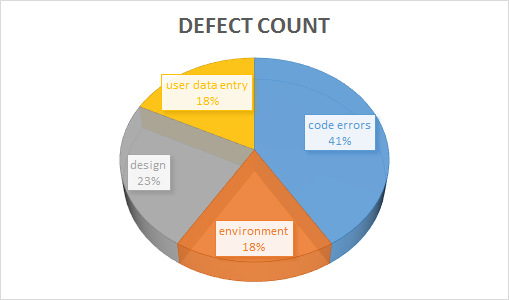
* *Code errors* represent the errors in the code cause defect.
* *Environment* means the defect in browser compatibility or system compatibility.
* *Design* part means our structure design is not very strict and trigger defects.
* *User data entry* defect represents wrong implementation in the database part.

So this graph is the number of defects in each type over the total number of defects during our project.

Until Iteration 2:



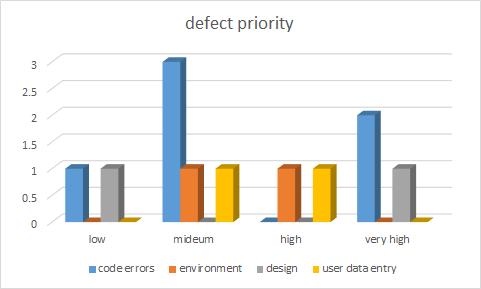
Final:

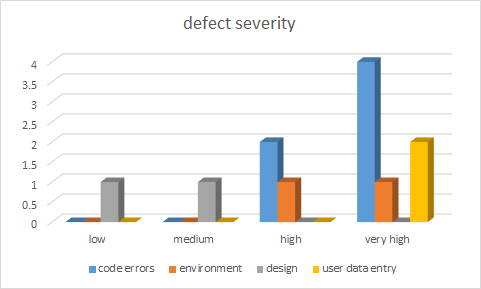


Graph 3 and Graph 4:

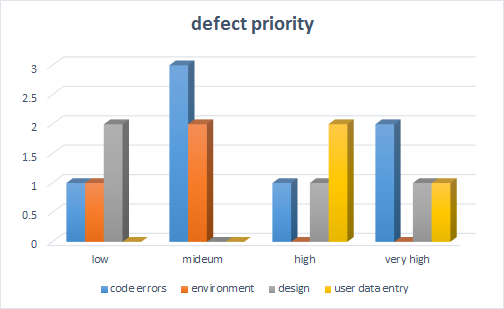
We have two basic criteria of defect management: defect priority and defect severity. Both of them have four levels: low, medium, high and very high. And the following two graphs represent the defect number in each type based on different defect management criteria.

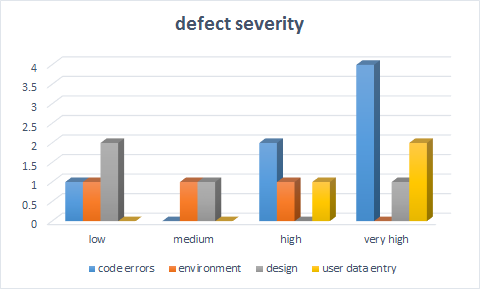
Until iteration 2:





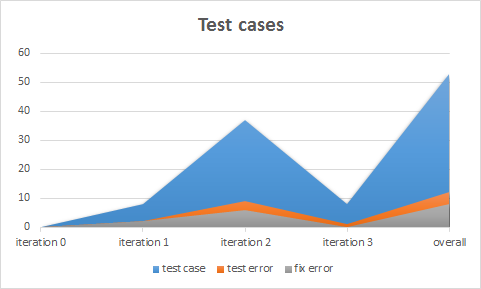
Final:





Graph 5:

The following graph represents the number of test cases in each iteration, the number of test error in each iteration and number of fixed error.



# References ( All)

# Glossary

Presentation : <https://docs.google.com/a/bu.edu/presentation/d/1OncXtilBObhzWB54klBlGucz1yeabmzAWt1fTXEAkRA/edit?usp=sharing>