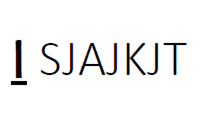
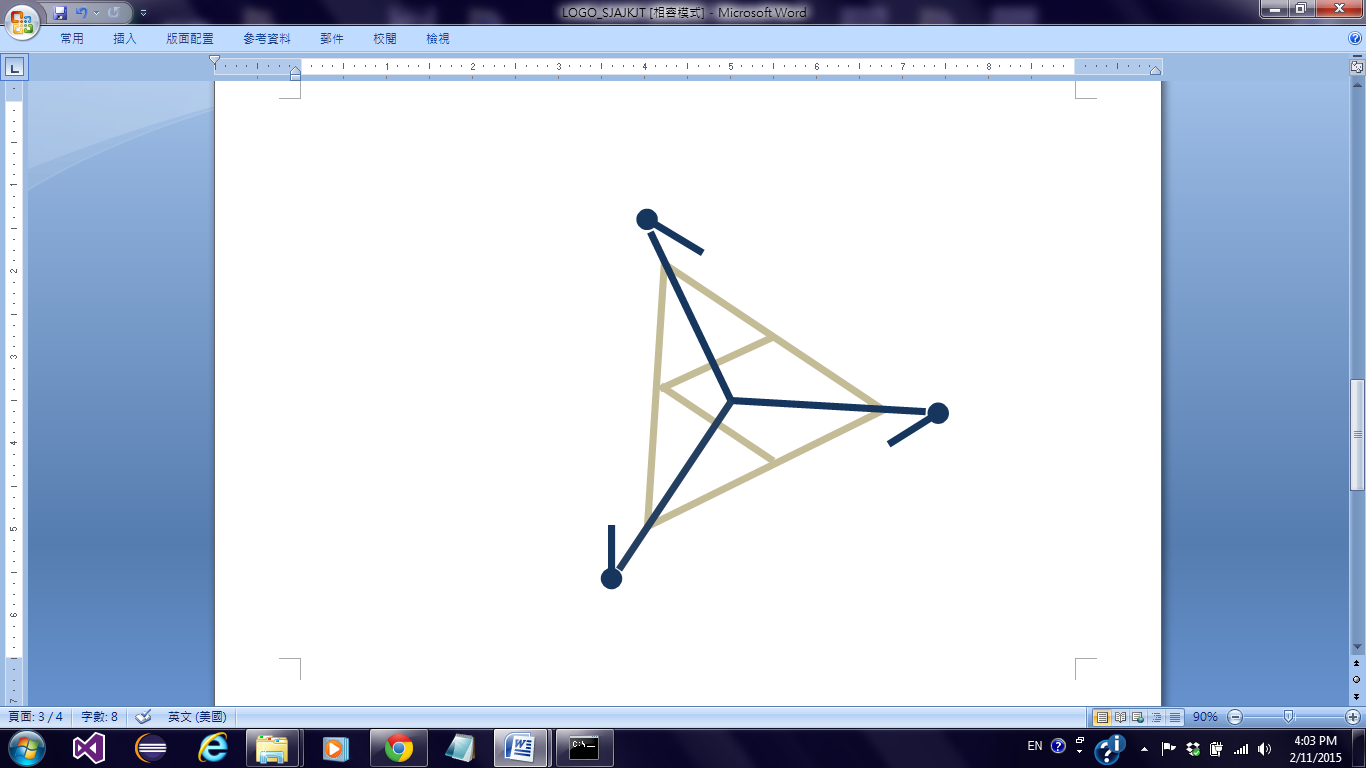
**CS673S15 Software Engineering** 

**Project 13 - SJAJKJT**

**Project Proposal and Planning**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Joe Driver | Team Lead |  |  |
| Joshua Darrieulat | Configuration Lead |  |  |
| Lin-Kei Tseng | Design Lead | *Lin-Kei Tseng* | 2/15/2015 |
| Chun-Kai Huang | Implementation Lead | Chun-Kai Huang | 2/15/2015 |
| Samer Abu-Nasser | Environment and Integration Lead | *Samer Abu-Nasser* | 2/15/2015 |
| Jerrold Ansman | Group 1 Leader |  |  |
| Ya-Lan Tsao | Requirement Lead |  |  |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#h.87t9hln2vjz0)

[Related Work](#h.mps353x5ezyl)

[Detailed Description](#h.fg3z0hpd4q9v)

[Management Plan](#h.ds8oyr75pnh1)

[Process Model](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.27177f40uci)

[Risk Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.a4oqwntk3mw)

[Monitoring and Controlling Mechanism](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.ywdoc2clc9yt)

[Schedule and deadline](#h.tadq5mb0pici)

[Quality Assurance Plan](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.72e1f4uawy2r)

[Metrics](#h.b2haznn3yyz2)

[Standard](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.vc72k6dweldv)

[Inspection/Review Process](#h.f1c69ifi68h7)

[Testing](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.r5d5mhtlf0kq)

[Defect Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.54a4wuncjg1c)

[Process improvement process](#h.jhct37ebxxpn)

[Configuration Management Plan](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.hw41vg4ykxen)

[Configuration items and tools](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.bwlb4d4vdox2)

[code commit guidelines](#h.yyauft6zr9hw)

[References](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.8mva2050iy7t)

[Glossary](#h.ty3i2nqffhtc)

# Overview

# (motivation, the purpose and the potential user of the software system etc. )

# Issue management tool to manage the issues or bugs in the project.

# **Motivation:**

# Develop one platform for post, follow, and solve problem effectively and efficiently.

# **The purpose and the potential user of the software system:**

# Users can post the bug issues they found immediately online to help developers find problems of their system efficiently.Besides,tracking and providing bug solutions.

**The potential user of the software system:**

Guest-Client Side,Manager-Server Side

# 

# Related Work

# (describe any similar software systems, and the difference from them)

# BugZilla

# Using status and resolution tracking function into our website

* + Build individual view issue

# Mantis

# using status and resolution tracking function into our website

* + Build individual view issue

# Redmine

* + Multiple projects support

# Trac

# JIRA

* + Clearly and friendly graphical user interface

# lighthoustapp

* + Create, edit and delete feature
  + List and sort issue clearly

# Basecamp

* + Almost everywhere can make comment which makes communication easier.
  + It can assign issue to one teammate.

# Bugify

* + Easy to set priority

# 

# Proposed High level Requirements

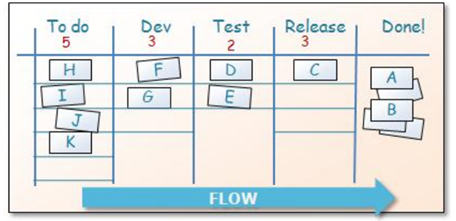
* 1. **Functional Requirements**
     1. **Essential Features**
* Report issue
* All members from the project can file ,create ,view ,and edit a defect report.
* Issues are analyzed based on several feature. (such as severity, importance, type, priority)
* History Tracking
* Status and resolution tracking
* Support for multiple databases
* Dashboard
* Login Account
* Milestone setting
* Issue/Bug Sorting
  + 1. **Desirable Features**
* Customizable email reports and notifications
* Search
* Built in wiki
* Issue calendar
* Tasks tracking
* Issue creation via email
* Project forums
  + 1. **Optional Features**
* mobile version website
* Attaching a file by dropping in the edit textbox directly
  1. **Nonfunctional Requirements**
* Security
* Color coding
* Running well in different browser

* 1. **Implemented Features (to be completed at the end of project)**

# Management Plan

# (For more detail, please refer to SPMP document for encounter example)

## Process Model

* + 1. The team will be using the kanban process model, a stage workflow process. This model will be implemented using the pivotal tracker tool, management documents and weekly meetings.
       1. 
    2. The team will decide next steps during each weekly sprint within the course-defined monthly iteration deadlines(3 iterations in total - approx 12 sprints). The next tasks for each iteration will be decided as a team and the tasks assigned according to experience, time and contribution effort. The team leader will monitor and manage the tasks put on pivotal tracker.
       1. The tasks will be split amongst the team members according to expertise and workload.
       2. Tasks are given a deadline.
       3. Milestone tasks are marked at the end of each sprint.

## Objectives and Priorities

* + 1. The objectives of the project are as follows
       1. complete the project by the project due date
       2. complete the project without straining a single resource
       3. provide all deliverables by the project due date
       4. Fulfill all requirements of the client as they fall into one of the following categories.
          1. Issue tracking creation, modification and archiving (closing)
          2. Issue tracking reporting capabilities
          3. Issue tracking historical capability (last modified by and modified history)

## Risk Management (need update constantly)

* + 1. Team Members drop out of project, or are overwhelmed due to larger group tasks as well. It’s hard to plan for this, but there are two things we can do to minimize impact.
       1. Get ahead. Any loss of personnel translates to lost time in catchup, so the best thing we can do to fight this is set a schedule that gives us room for error...and time to catchup from losses.
       2. Internal partnering. We need to do our best to make sure no single person completely owns any single aspect of the project. In the event we lose someone, there needs to be a second person who can step into their role fairly quickly.
    2. Onboarding issues: Quite a few team members entered without knowledge of systems like git, django, or the python language. A major associated risk is allowing people to fall behind on the learning curve (i.e., making someone play catch-up for the whole semester). To combat this we spent the first week focused on onboarding, emphasizing tutorials, useing code, etc.
    3. co-dependent products: Since we are working on a feature that provides several interfaces to information that is all derived from a single source, we could encounter a situation where several team members are stuck waiting for someone else to complete their work before they can proceed. This translates to lost productivity. To approach this issue we’d like to see a situation where our teammates are not married to a single task. For example, if the testing lead cannot design a test for an interface because the interface is not yet functional, it would make sense to reassign him or her temporarily to support for the interface or to another feature entirely. To make this possible, the testing lead cannot have been soley focused on testing up to that point, a diverse understanding of our project up to that point is necessary.

## Monitoring and Controlling Mechanism

* + 1. Currently using a mixture of Slack, Pivotal Tracker, and google docs. It is the team lead’s responsibility to keep abreast of each individual’s status. Slack is monitored as often as possible so other teammates can respond to appropriate issue, and pivotal tracker should ensure that work is always available to those who need it.

## Schedule and deadlines (need update constantly)

* + 1. Currently running bi-weekly meetings where we assess and schedule deadlines. The short span between them allows quick adjustment, and means that missed deadlines don’t put us too far back (allows us to allocate appropriate resources in time). Weekly schedules are kept in google docs and updated with each meeting.

# Quality Assurance Plan

Quality Assurance means that all those planned and systematic activities implemented to provide adequate confidence that an entity will fulfill requirements for quality.

## Metrics

* + 1. **Definition** (e.g. define what metrics will be used, , how to keep track of metrics, and how to analyze the metrics for process improvement. Two types of metrics should be included: product metrics and process metrics. Particularly include product complexity (LOC, # of files, # of classes, # of methods etc.) cost (in terms of man hours), defect and defect fix rate etc.

Metrics can help us to measure software quality level , estimate project schedules , track schedule progress, measure software size and complexity,determine project cost, and process improvement

**Product metrics**(Quality Control):

product complexity: Size and Time Cost

Size:

How many Lines of Code in our the methods, class, files of issue tracking system? How large Database? How big a attachment file in one issue report?

Cost :

How long time spend? How many people involve in one subproject?

Defect :

How many bugs in the code on test process?

Coverage:

How much of the project is adequately covered by testing?

Defect fix rate :

How many issue resolution we complete?

Customer problems:

Count of user complain,User turnover rate

Customer satisfaction:

User experience and professor advice

**Process metrics**(Quality Assurance):

Project effort:

7 members in our issue tracking project on 3 months

Defect repair rate:

Check the defect repair rate on team3 Github Graphs

Productivity:

Check how many tasks are finished on time in our Pivotal Tracker system.

* + 1. **Results** (to be completed at the end of each iteration)

## Standards and tools

* + 1. **Coding Standard:**

Django provides most of what we’ll likely need in regards to building tests, although if we decide to implement mocks and patches we may need to look elsewhere (BeautifulSoup?). Selenium can be used for front-end/black box testing and is supported by django.

Coding standards can be further enforced by a regimented deploy process to our master branch.

* + 1. **Tools:**

Django 1.7.4

coverage 4.0a4

pep8 1.6.1

pyflakes 0.8.1

pylint 1.4.1

mock 1.0.1

Fabric

Code Pro (Python) - standardize every line of code

iii. **Documentation Standard:**

1. The document standard on Blackboard,Google drive is followed by professor’s description.
2. The document standard of Github and Pivotal Tracker is followed by team memebers’ discussion.

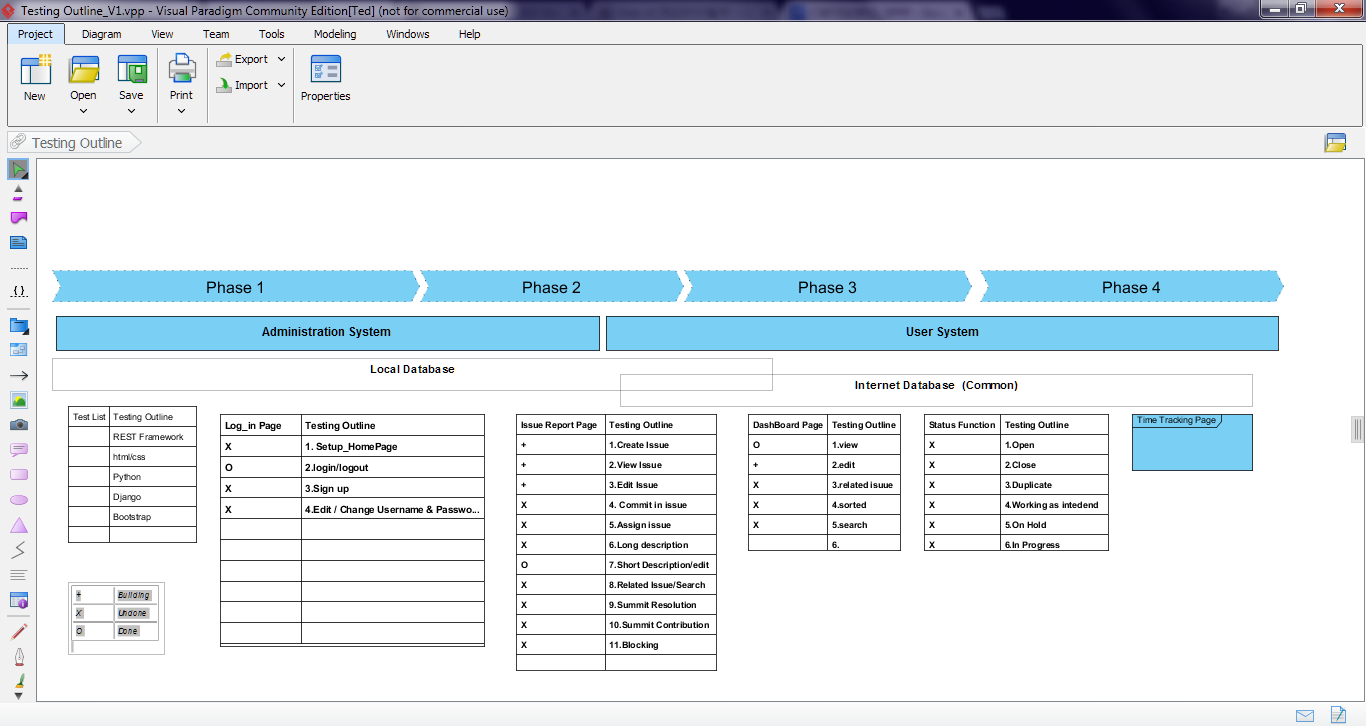
## Inspection/Review Process

* + 1. Anything pushed to production should first see a code review by someone who has not worked on the feature. At this stage a part of the review should cover whether or not we have appropriate unit testing in place for this feature. Once it is approved at that stage, it should be manually tested by whomever will be writing the selenium test for it in a staging environment (this way the selenium tester will have a better idea of how to approach their test cases). Once approved on both fronts it can be deployed to production with selenium testing to follow.

## Testing-Bottom-up Testing

## Because most features will be define at beginning, Bottom-up Testing Strategy should be a choice. Bottom-up Testing Strategy is an approach to integrated testing where the lowest level components are tested first *(ex: Layer 1 from A~G)*, then used to facilitate the testing of higher level components *(ex: Layer 2 from H~J)*. If the one of component on *layer 2* has a bug, the tester will back to lower layer,*Layer 1*, to test and debug again. The process is repeated until the component at the top of the hierarchy is tested*(ex:Layer 4, M)*. When a unit finish, the person who build the program will test and fix it. A higher layer will be tested every iteration, and all action will repeat until the whole project finish.

A separate document about testing result should be linked here.

***Testing Outline:*** 

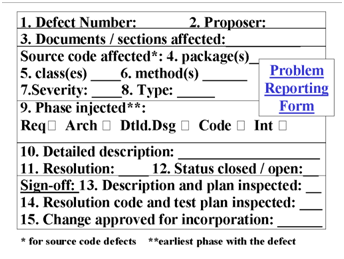
## Defect Management

**The criteria of defect:**

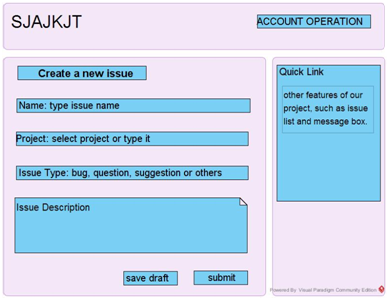
Defect need to be tracked,fixed, and managed during all phases of the issue tracking project.

1. Deviation from customers requirements of issue tracking system
2. Severity:Major, Medium, Trivial
3. Priority: 0~5 point
4. Type: inconsistency,unnecessary,omission,unclassified

Basic defect report form :



Issue report mockup:



## Defect Management Process

Bug reports should go through a process that ensures that they aren't closed if they shouldn't be. Such a process should require that the person who reported the bug be the one to verify that it has been fixed or to approve any other type of resolution.



[Defect Prevention --> Deliverable Baseline -> Defect Discovery -- >Defect Resolution -->Process Improvement -->Management Reporting]

**Defect management tool we use:**

* **Github -** (manage defect):

Checking issue tracking system mockups we design and HTML-mockups

* **Pivotal Tracker -** (manage action)**:**

Tracking every user story, man hours, and story priority.

* **Slack & Google Handout Chatting Tool -** (manage personnel):

Discuss and share team members’ idea ,and point out any problem instantly.

(Still under consideration by QA leads.)

# Configuration Management Plan

(For more detail, please refer to SCMP document for encounter example)

## Configuration items and tools pep8 - python formatting pyflakes - python error checking. unit tests - Provided via django.

## Change management and branch management Given the small size of our team, developers can choose to commit to master or work on a branch and then do a merge to the master branch. For larger feature work, it is recommended to use a branch. Individual commits should be small and never break the application. Any commit that breaks unit or integration tests will be rolled back.

## Code commit guidelines All code should pass unit tests (code validation), pyflakes(python error checking), and pep8 (python formatting rules). Any new python code must include unit tests with the code.

# References

(For more detail, please refer to encounter example in the book or the software version of the documents posted on blackboard. )

<http://wiki.mbalib.com/zh-tw/%E6%95%8F%E6%8D%B7%E5%BC%80%E5%8F%91>

<http://code.tutsplus.com/articles/the-principles-of-agile-development--net-25732>

<http://androchen.logdown.com/posts/2014/03/19/productivity-carpo-development-project-management-tools>

<http://www.slideshare.net/littlebtc/git-5528339>

# Glossary