****

CS673S15 Software Engineering

Group 1 Team 3 Project

Project Proposal and Planning – Issue Tracker

**Team**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Member** | **Role** | **Signature** | **Date** |
| Joe Driver | Team Leader/QA Leader | Joe | Feb-15-2015 |
| Ya-Lan Tsao (Amy) | Requirements Leader | Amy | Feb-15-2015 |
| Lin-Kei Tseng (Ted) | Design Leader | Ted | Feb-15-2015 |
| Chun-Kai Huang (Kenny) | Implementation Leader | Kenny | Feb-15-2015 |
| Jerrold Ansman | Group Leader |  | Feb-15-2015 |
| Samer Abu-Nasser | Environment and Integration Leader | Samer | Feb-15-2015 |
| Joshua Darrieulat | Configuration Leader | Josh | Feb-15-2015 |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| 1.0 | Joe Driver | Feb-15-2015 | Kick-off |
| 2.0 | Team | April-2-2015 | Iteration 2 |
| 3.0 | Team | April-21-2015 | Iteration 3 |
| 4.0 | Samer Abu-Nasser | May-3-2015 | Final |

Contents

[Overview 3](#_Toc418505958)

[Related Work 3](#_Toc418505959)

[Proposed High-level Requirements (Implemented features in Green) 4](#_Toc418505960)

[Functional Requirements 4](#_Toc418505961)

[Non-Functional Requirements 4](#_Toc418505962)

[Management Plan 4](#_Toc418505963)

[Process Model 4](#_Toc418505964)

[Objectives and Priorities 5](#_Toc418505965)

[Risk Management 5](#_Toc418505966)

[Qualitative Risk Assessment 6](#_Toc418505967)

[Risk Assessment Matrix 7](#_Toc418505968)

[Monitoring and Controlling Mechanism 7](#_Toc418505969)

[Schedule and deadlines 7](#_Toc418505970)

[Quality Assurance Plan 7](#_Toc418505971)

[Metrics 8](#_Toc418505972)

[Product and Test Metrics 8](#_Toc418505973)

[Process Metrics 8](#_Toc418505974)

[Standards and Tools 8](#_Toc418505975)

[Coding standard 8](#_Toc418505976)

[Tools 8](#_Toc418505977)

[Documentation Standard 9](#_Toc418505978)

[Inspection/Review Process 9](#_Toc418505979)

[Testing: Bottom-up Testing 9](#_Toc418505980)

[Defect Management 9](#_Toc418505981)

[Defect Management Process 10](#_Toc418505982)

[Defect Management Tools 10](#_Toc418505983)

[Defect Management Plan 10](#_Toc418505984)

[Change Management 10](#_Toc418505985)

[Code Commit Guidelines (Master level) 10](#_Toc418505986)

[References 10](#_Toc418505987)

[Glossary 11](#_Toc418505988)

# Overview

Team 3 was assigned to create an issue management tool to manage issues or bugs in the project.

**Motivation**: Develop a platform to allow project teams to post, update and resolve bugs and issues effectively and efficiently.

**Purpose and Potential Users**: Project Managers, Developers, Testers and other stakeholders can view the issues and bugs related to their project, create new ones and follow up on specific issues as needed. This tool will help users *“Log, Track, Resolve and Ship. Issue Tracker provides a simple interface to help get your app out of crap.”*

# Related Work

* Bugzilla
  + Relevant Similarities
    - status and resolution tracking
  + Relevant Differences
    - Build individual view issue
* Redmine
  + Relevant Similarities
    - Status and issue tracking
  + Relevant Differences
    - Multiple projects support
* Trac
  + Relevant Similarities
    - Status and issue tracking
  + Relevant Differences
    - Wiki - Trac
    - Dashboard and Status counts – issue tracker
* Lighthouseapp
  + Relevant Similarities
    - Create, Edit Features
    - List & sort issues
  + Relevant Differences
    - Delete Feature – Lighthouseapp
* Basecamp
  + Relevant Similarities
    - No separation of duties
    - Issue Assignment
  + Relevant Differences
    - Verify process – issue tracker
* Bugify
  + Relevant Similarities
    - Priorities

# Proposed High-level Requirements (Implemented features in Green)

## Functional Requirements

* 1. Essential Features
     1. Report Issue
     2. All members can create, view and edit issue report
     3. Issues are sorted based on attributes (severity, importance, type and priority)
     4. Historical tracking
     5. Status and resolution tracking
     6. Support for multiple databases
     7. Dashboard
     8. Login Account
     9. Issue/Bug Sorting
  2. Desirable Features
     1. Milestone relations
     2. Customizable Email reports and notifications
     3. Search
     4. Built in wiki
     5. Issue Calendar
     6. Tracking (Overview counts assigned per user, resolved per user…)
     7. Project forums
     8. Email plugin
  3. Optional Features
     1. Mobile version
     2. File attachments

## Non-Functional Requirements

1. Security Features (Access control, Password Authentication, Allowed URLs)
2. Color coding
3. Cross-Browser support

# Management Plan

## Process Model

The team will use a Kanban process model – an agile and iterative approach. The model will be implemented using the pivotal tracker tool, management documents and weekly meetings.

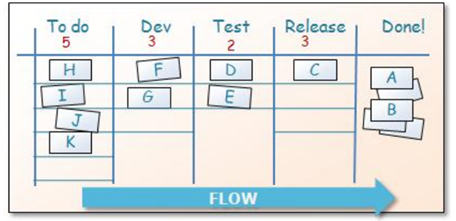


Figure 1 - Kanban Process Diagram

The team will decide on the next features or tasks during at the weekly sprint kickoff meeting. Team 3 will go through approximately 12 sprints spread out amongst the 3 course-defined iterations (approx. 4 sprints per iteration).

## Objectives and Priorities

The objectives of the project are as follows:

1. Complete project by due date (Apr-29-2015)
   1. Integration included
   2. Feature completion two weeks before
   3. Final testing one week before
2. Complete project without straining a single resource
   1. Provide all deliverables by the project due date
   2. Fulfill all requirements of the client as they fall in to one of the following categories
      1. Issue tracking creation, modification and archiving (closing)
      2. Issue tracking reporting capabilities
      3. Issue tracking historical capability (filter by attributes and status)

## Risk Management

|  |  |  |
| --- | --- | --- |
| **Rating** | **Likelihood** | **Definition** |
| 1 | Rare | Occurs in exceptional circumstances |
| 2 | Unlikely | Could occur at some time |
| 3 | Possible | Might occur at some time |
| 4 | Likely | Will probably occur in project |
| 5 | Very Likely | Expected to occur in most situations |

Table 1- Definition of risk Likelihood values

|  |  |  |
| --- | --- | --- |
| **Rating** | **Likelihood** | **Definition** |
| 1 | Insignificant | No cost and/or schedule impact |
| 2 | Minor | Minor cost and/or schedule impact |
| 3 | Moderate | Significant cost and/or schedule impact |
| 4 | Major | Extensive cost and/or schedule impact |
| 5 | Catastrophic | Unrecoverable cost and/or schedule impact |

Table 2 - Definition of risk Impact values

### Qualitative Risk Assessment

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Risk ID** | **Category** | **Risk** | **Likely-hood** | **Impact** | **Mitigation Action** | **Contingency Action** | **Opportunity** | **Outcome** |
| **1** | Time | Completion: Project is not completed on time | 2 | 4 | -Prioritize features by simplicity and criticality and complete those with high ratings | - Drop desirable and optional features | Increase quality of core functionalities | Project was completed on time |
| **2** | Resources | Personnel loss - Team members drop out or are overwhelmed | 1 | 5 | - Break down tasks into smaller components  - Try to get ahead | - Reduce the scope of the issue tracker application.  - More experienced members take on more tasks | Team members gain more experience in Django and software in general. | No members were lost |
| **3** | Process | Onboarding - many team members without knowledge/lack of experience in core toolset (git, python, Django…) | 4 | 3 | Hold training sessions and pair experienced team members with non-experienced members | -Experienced programmers focus on development rather than other tasks | Team members get a broad knowledge and experience in new tool environment | Team members better than expected, however at a loss of a better product, accepted. |
| **4** | Integration | Co-dependent products may cause integration risk in putting the project together. Such as project and user models and the login feature. | 3 | 3 | Cross group meetings to discuss the structure and analyze compatibility issues and design around these issues | Static Fields and the apps will work as completely separate non-shared data applications. | Team members get a sense of integration work and work with other teams on different applications. | Integration went well, no need for separate application, planning ahead strategy worked after hard work. |

### Risk Assessment Matrix

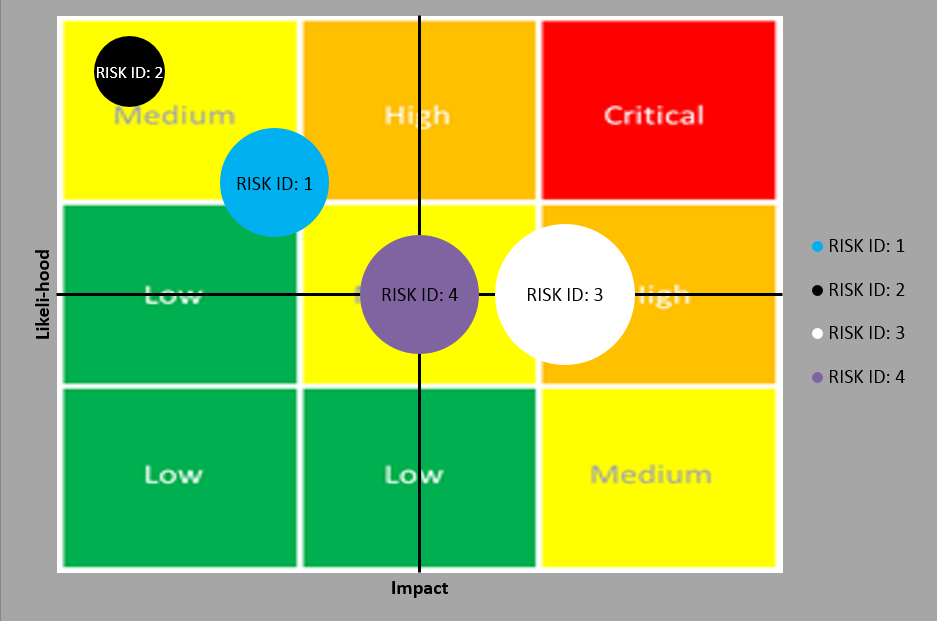


Figure 2 - Risks Map

## Monitoring and Controlling Mechanism

Slack, Pivotal Tracker and google docs will be used to monitor and control the project. The Team Leader will keep abreast of each individual’s status.

Slack will be the primary tool to keep up with teammates and adjust sprint tasks. Pivotal tracker will be the base line of our goals or user-stories. The risk register (and matrix) above will be monitored and adjusted throughout the project.

## Schedule and deadlines

Weekly meetings assess and schedule deadlines using the management process specified in the prior process section. The sprints allow quick adjustment and means that missed deadlines due to technical or scheduling difficulties do not impact the project path. Weekly schedules are kept in google docs and updated with each meeting in the minute documents.

# Quality Assurance Plan

Quality assurance identifies all the planned and systematic activities implemented to provide adequate confidence that an entity will fulfill requirements for quality.

## Metrics

### Product and Test Metrics

#### Size and Test

The size of the project will be measured according to database size and records.

#### Cost

The cost of the project would be determined by man-hours. In the weekly report document, team members would fill out what they did that week in terms of tasks and the respective amount of hours spent.

#### Coverage, Bugs and Fixes

The project test coverage (also see Testing in the Software Design document) will be done manually, and bugs fixed before code commit. Given the size of the project, heavy testing will not be required. However, it is noted that with increase in the size of the project, the team will consider a Test-Driven-Development approach as the Object based test design was extremely helpful in the code process. Our features (listed in the requirements section) were to be verified manually after each feature submission and applicable functional tests run. This will be done once as submitted by the team member during staging submission and again in aggregate when a final push to the master branch of all new stating features are required.

### Process Metrics

#### Effort

7 Team members will be working the issue tracking project for 3 months.

#### Defect Repair Rate

The teams code standards, configuration and commit process was very successful. In fact, only one defect was found (Edit issue returns 404). Otherwise our development and testing process really shined. The defect was found and was resolved by the end of the next sprint. In fact, it was not a defect but a requirements issue that had to be adjusted.

#### Productivity

The team was able to complete all required features and added in optional and desirable features as well. Many of the team members were novice web developers and even new to the tools being used. Our team was very productive and given this was a course contribution, the most important thing was that every member walked out with at least 2 courses worth of knowledge at the end of the course.

## Standards and Tools

### Coding standard

Django/Python in combination with selenium will cover our testing needs. Plugins such as Pep8 (and are listed in the tools section below) are being used to ensure syntactic accuracy and conformity.

### Tools

1. Coverage: code coverage tool
2. Pep8: python formatting
3. Pyflakes: python syntax error checking
4. Pylint: python convention checking
5. Unit test: unit testing library provided by Django & Python

### Documentation Standard

The Documentation standard will be put into git hub, while high-level tasks are reviewed and tracked in pivotal tracker. Finally, miniature tasks are held in “stand-up” sprint meetings.

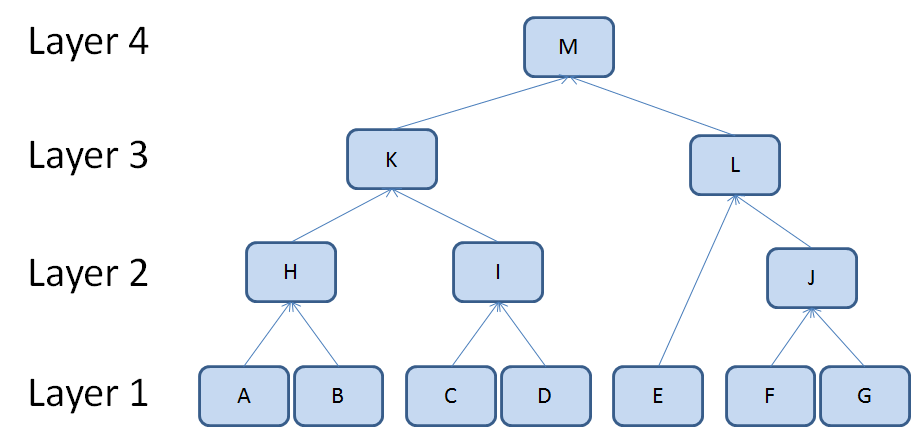
## Inspection/Review Process

Production codebase updates are done with a check and balance system. IT is pushed to the staging branch then reviewed by the Quality leader for merging at the team level. At the group level, branches for each team are created in a pre-production and production fashion. Dates are agreed upon for the production branch to be pushed to the master by the Environment and integration leader post testing and quality tests.

Testing will be done manually and selenium tests will be run for assurance, the testing system was not optimized due to the lack of experience of the team in the general environment. The team relied on manual testing and provided selenium functional tests.

## 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.



## Defect Management

Criteria: An issue or bug that violates the functional tests or causes an application or system error.

The following list will classify a defect for the projects purposes.

1. Deviation from Customer requirements
2. System impact
3. Application impact
4. Priority

### Defect Management Process

Bug reports should go through a process that ensures bugs aren’t closed without resolution or reasonable cause. Bugs will be created in the bug log and resolved as needed. The team aims for a zero-bug process methodology by heavy functional testing and given the size of the project a full functional review before moving to the staging and product code base.



### Defect Management Tools

1. GitHub
   1. Regressions and Deployment Bugs
2. Pivotal Tracker
   1. New requirements that arise as a result of bug or issue
3. Slack
   1. Discuss ideas and reach out to team members on any problems for help

## Defect Management Plan

### Change Management

Given the small size of the team, developers will commit to a staging branch. A merge will be done by the Quality leader after the necessary tests are done to ensure a strong code base that is ready for integration at the master level. The master level maybe 10-20 commits behind the staging branch at a single time. Any commit that breaks the tests or any facet of the application will be rolled back.

### Code Commit Guidelines (Master level)

* Code should pass testing and quality control process
* Code should adhere to coding standards and formatting
* Code should include the necessary tests to allow for ease of development.

# References

<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