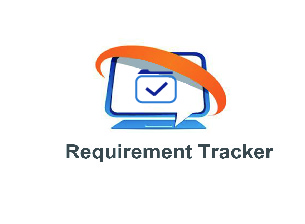
**CS673S16 Software Engineering** 

**Team II - Requirement Tracker**

**Project Proposal and Planning**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Rui Dai | Team Leader Requirement Leader | *Rui Dai* | 2/10/2016 |
| Nora Salamah | Backup Team Leader  Requirement Leader | *Nora Salamah* | 2/10/2016 |
| Ruirong Yang | Design Leader | *Ruirong Yang* | 2/10/2016 |
| Chengchen Wang | Environment and  Integration Leader | *Chengchen Wang* | 02/10/2016 |
| Divyam Hansaria | Implementation Leader  QA Leader | *Divyam Hansaria* | 02/10/2016 |
| Minteng Xie | Security Leader | *Minteng Xie* | 02/10/2016 |
| Zhi Dou | Security Leader | *Dou Zhi* | 02/10/2016 |
| Abhyudaya Alva | Configuration Leader  Implementation Leader | *Abhyudaya Alva* | 02/10/2016 |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| 1.0 | Nora Salamah | 2/5/16 | Add |
| 1.1 | Chengchen Wang | 2/10/16 | Add Section |
| 1.2 | Rui Dai | 2/10/16 | Change Format |
| 1.3 | Nora Salamah | 2/23/16 | update section |
| 1.4 | Divyam Hansaria | 4/10/16 | Update QA Plan Section |
| 1.5 | Minteng Xie | 4/21/16 | Modify security part |
| 1.6 | Nora/Rui | 4/21/16 | Modify requirements & management plan parts |
| 1.7 | Chengchen Wang/Ruirong Yang | 4/27/16 | Modify requirements parts and test parts |
| 1.7 | Abhyudaya Alva | 4/27/16 | Product and Test Metrics |

[Overview](#h.g6igqliy7rm)

[Related Work](#h.bf21eadgjj29)

[Proposed High level Requirements](#h.rgyo4hi9stmq)

[Management Plan](#h.ts358bsdtbcv)

[Process Model](#h.b9kfvy9p3hab)

[Objectives and Priorities](#h.83nauhq6ogqy)

[Risk Management (need update constantly)](#h.gimqjrurknm2)

[Monitoring and Controlling Mechanism](#h.jgtvk2lib7aa)

[Schedule and deadlines (need update constantly)](#h.352qa75o3dhf)

[Quality Assurance Plan](#h.quttytfw209u)

[Metrics](#h.ga1i3jbjlzek)

[Standard](#h.w394mxlvrxyd)

[Inspection/Review Process](#h.x7qdaln2b54l)

[Testing](#h.588qbzpexrrt)

[Defect Management](#h.cfxi1l6fp7lj)

[Configuration Management Plan](#h.j5uvivmxqcsp)

[Configuration items and tools](#h.zdj31vlso4l6)

[Change management and branch management](#h.octvwbz18y0r)

[Code commit guidelines](#h.qz6clpvim2zx)

[Security Process](#h.ty3i2nqffhtc)

[References](#h.pydogwn1s4ot)

[Glossary](#h.ynailbiil7nv)

# Overview

The project aims to redesign the Requirement Tracker in 3blueprint.com to help organization leaders have a clear real-time view of the internal operation of the project and make an alternative plan immediately when problems occur or for better productivity. According to the existing Requirement Tracker, some bugs should be fixed and more essential features should be extended to make the tool more comfortable to use. By automatically merging alignment of strategic plans, project plans and daily to-do lists into a big picture, project managers can keep track of progress and guide the team work. At the same time, project team members have a clear view of their tasks. The potential users are people in the whole company from top managers to developers. The Requirement Management software is based on an agile model.

# Related Work

3BluePrints was first established as a project by a previous software engineering team and handed over the beta version to us. It is already a good beta version that can be used for basic projects, however there are still lots of bugs and half-baked functions. So our team’s goal in the first iteration is to understand the existing system, test the already created tests and create new ones. The second iteration will focus on fixing the bugs and then complete the unfinished functions, while we are also planning to add some useful features to this system by the second iteration. The beta version was developed using Python, Django framework and SQLite, which was simple and neat, so we decided to work on this framework too, except some adjustment on SQLite. And of course this project is an open-source project.

There are already several similar software systems for developers, we have done some research on those products and listed their features and the differences between those and 3blueprints. Below are several main differences between 3BluePrints and other commercial or open-source tools:

1. Team City: Jet Brain’s team tool Team City is an open-source free project management tool. It’s an integration of issue tracker, code review and project management tools. It’s a web-based tool and perfectly compatible with Github. You can connect to Github by just entering repository URL in the tool. And this is what 3BluePrints can’t do for now. And for each project, Team City provides statistical chars like bullet charts and pie charts that can be easily viewed. This is what we want to add into 3BluePrints. Besides, Team City provides connections to Cloud platforms. And the best feature is you can get outputs of the project via this tool, so it’s convenient to produce documentations during developing the project.

2. Pivotal Tracker: This is a famous commercial project management tool, although it’s free for small startups. As our members thought, Pivotal Tracker is considered a story-based project management tool that allows team members to work on software development project easily, while it is very user-friendly. At least for now, 3BluePrints is neither user-friendly nor easy to get start to use for most team members. Compared with Pivotal Tracker, the user logic of 3BluePrints need to be revised so new users don’t need a lot of time to know how to use everything. But at least, 3BluePrints provides a way (chat room) for team members to communicate with each others.

3. Active Collab: This is also an agile project management tool that is user-friendly and suitable for all customers from personal to large companies. It can create multiple tasks, and it provides “My Work” function that each user can check their current work, activity history and time log with ease. What’s more it provides money tracking and calendar views, which are very useful to reduce the cost for a team. These are features that 3BluePrints lacks of.

4.Jenkins: This is an open source continuous integration tool written in Java. It is used for testing and reporting on isolated changes in a larger code base in real time. It enables developers to find and solve defects in a code base rapidly and to automate testing of their builds. This is very helpful when running tests and merging & tracking changes in Git.

Beside, the above four tools, there are a large number of tools, but their main features are similar to each other. If we cannot add some very cool and indispensable functions to 3BluePrints, the most important task is to fix bugs of 3BluePrints and redesign the interface, and complete the unfinished functions.

# Proposed High level Requirements

**Current features**

1. **Essential Features**
2. A new User should be able to sign up for an account.
3. A user should be able to log in to the system if they have an account.
4. An admin should be able to assign various roles for different users.
   1. A user should be able to view only functionalities of his role.
      1. Owner:
   2. Owner should be able to view & create & edit & delete projects.
   3. Owner should be able to change its role from owner to developer or client. Once the role is changed to developer or client, one can no longer change role.
   4. Owner should be able to create & edit & delete user story in icebox. can add task in user story, set & change story owner, set hours, status, scale points.
   5. Owner should be able to add & edit & delete comments
   6. Owner should be able to create/edit/delete iteration
   7. Owner should be able to add & edit & delete task in each iteration.
   8. Owner should be able to upload & download documents.
   9. Owner should be able to update status of story to started/completed.
   10. Owner should be able to assign roles to users and add them to the project.

ii. Developer:

1. Developer should be able to view projects
2. Developer should be able to create & edit & delete user story in icebox.
3. Developer should be able to add task in user story, set & change story owner, set hours, status, scale points.
4. Developer should be able to add & edit & delete task in each iteration, no matter who create the task
5. Developer should be able to add & edit & delete comments
6. Developer should be able to upload & download documents.

iii. Customer:

* + Customer should be able to view projects
  + Customer should be able to create & edit user story
  + Customer should be able to add & edit & delete comments

**New Features**

* + 1. **Functional features**
       1. **Essential**
          1. A User story should not be less than 0 hours.
          2. A project name should not be duplicated.
          3. A drop down list containing all roles available should be available for customers during sign up.
          4. A project can not have no owner at any time.

Project owner can change its role from owner to developer or client. Once the role is changed to developer or client, one can no longer change role.

* + - 1. **Desirable**
         1. User receives email regarding any changes to accou

A user should receive a confirmation email regarding any changes to his/her account.

* + - * 1. User name on comments

A user name should be displayed with the comment entered.

* + - * 1. No iteration warning

When there are no iterations available an error message should be displayed when trying to move a user story.

* + - * 1. Generate project report

A report should include all project related information (project name, user stories, iterations, tasks).

Report is in pdf format.

* + - 1. **Optional**
         1. Budgeting of project (financially)

Owner should able to add budgets to the task

Owner should see a report about the money use. This would help the owner to make changes if the task is too costly.

* + - * 1. Gantt chart

Owner/Developer should able to see the Gantt chart to know the priority of the task

* + - * 1. Labeling feature

Owner/Developer should be able to add labels to the story

Owner/Developer/Client should be able to filter stories based on label.

* + 1. **Nonfunctional features**
       1. Database remodeling.
       2. Constraint of iteration start date ( can’t be earlier than current date)
       3. Improve the User interface
          1. IceBox

link should be isolated from iterations ( should be moved under Project details).

Adding User stories to IceBox should be in the IceBox Page.

* + - * 1. Each iteration should have two sections for the stories (Current and Done)
        2. adding Home link, logout link
      1. Security
         1. Lock account for 1 minute if user fails to login 3 times.
         2. Log out automatically if page is closed.
         3. Logout if there is no activity for 10 minutes.
    1. **Implemented features**

|  |  |  |
| --- | --- | --- |
|  | **Feature** | **Functional/non-functional** |
| **essential** | Constraint on iteration start & end date. | Non-functional |
| Lock account for 3 unsuccessful login | Non-functional |
| No duplicate project name | Functional |
| User story hours can’t be 0 | Functional |
| Add role drop down list in sign up page | Functional |
| Automatically logs out when page is closed | Non-functional |
| **desirable** | Time-out handler | Non-functional |
| User receives email regarding any changes to account. | Functional |
| User name on comments | Functional |
| No iteration warning | Functional |
| Generate project report in a complete PDF file | Functional |

# Management Plan

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

## Process Model

The process used in our project is going to be an agile approach. The project starts with a planning phase and submitting a SPPP document. Then the project will be done in 3 iterations, each ending by demonstrating a working system. Each iteration includes planning, requirement & risk analysis, coding & testing, monitoring. The project due date is April 27th, 2016 where a final demo of the project will be presented by the team.

## Objectives and Priorities

The objective of the project is to fix the bugs and enhance some features that already exist and fulfill the requirements documented in section 3. The plan is to achieve this objective and deliver a high quality system within the time frame. All team suggestion of requirements are added in the icebox in the pivotal tracker. The requirements are prioritized and assigned to members so they can start working on them and finish them by the end of each iteration.

## Risk Management (need update constantly)

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk ID** | **Risk Event** | **Impact of Risk** | **Risk Prevention Mechanism** |
| 1 | Absence of any of the project team members.(sick/dropped class) | Progress on the project will be jeopardized. | Have backup for each member. |
| 2 | Unexpected interruption such as snow days. | Progress on the project will be jeopardized. | Continue working on project as planned, have meetings online or through google hangouts. |
| 3 | Different experience level of team members. | Delay task completion and project. | Try to divide team to subgroups. Each having an experienced member to try to help the rest. |
| 4 | Iteration requirements or task not completed at end of iteration. | Progress on the project will be jeopardized. | Have a meeting and prioritize the requirements and plan next iteration accordingly. |
| 5 | Major conflict when doing the merge at the end of iteration | Some of the function can not run. The worst situation is that the project can not be recovered. | Each sub team has the backup of work. There should be a project backup for each iteration. |

**Requirement Assessment Matrix:**

## 

As seen above we have 1 low risk (2), 1 medium risk (1), and 2 high risks (3 and 4). The two high risks will have high impact on the project progress and should be addressed first as planned.

## Monitoring and Controlling Mechanism

Team will communicate and make sure they are on track and schedule through:

o Weekly meetings.

o Weekly reports.

o Minutes of meetings report.

o Slack communication.

o Pivotal Tracker.

o Google Docs.

o Subteam work.

## Schedule and deadlines (need update constantly)

The project will go through 3 iterations:

o 1st iteration is due by 3/2.

1. Understand current system
2. Test current system

o 2nd iteration is due by 3/29.

i. fix bugs

o 3rd & final iteration is due by 4/27.

i. Add new features

The monitoring and controlling mechanisms mentioned in the section above will help in tracking our schedule and measuring the success of each iteration.

# Quality Assurance Plan

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

## Metrics

We will be using two metrics to keep track of quality of work we are doing. Product and Test metrics and Process metric to see if the process we are using is effective.

* + 1. **Definition**

**Product and Test Metrics**

**Complexity:** complexity of the project is determined by the number of features enhanced or fixed, number of line of code(LOC).

**Size and Test:** The size of the project will be measured in terms of bugs fixed and improvements implemented.

**Cost:** Cost of the project is determined by the man-hours put by the team in the entire project. Man-hours are tracked by all team member’s input in the weekly report document which specifies how many hours are spent on particular task.

**Coverage, Bugs and Fixes:** Test Case Coverage Rates (number of modules/classes/methods covered by at least one unit test ), Test Case Pass/Fail Rate will be determined in product and test metrics.

**Process Metrics**

1. Deadlines for each iterations along with number of task decided (estimated completion time vs actual)

2. Resource usage and allocation(estimated hours to complete vs actual)

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

**Product and Test Metrics**

Total Number of Lines of Code Added to the Project : 3662

Total Number of Lines Deleted From the Project : 229

Total Number of Unit Test Cases: 58 Pass Percentage : 100%

Total Number of UI Test Cases : 13 Pass Percentage: 100 %

Complete report about all the test cases added and the unit tests have been uploaded to the SDD document. Please refer to the bel

[SDD Testing](https://docs.google.com/document/d/1snwXidIx6QS7r6JTz2p3oAPvvmtIL5ebPfk9loI2NZ0/edit)

Coverage report has been generated through the coverage plugin. The report has been added to team folder (Please refer to coverage\_report.xls ). The coverage report has been generated after running both the unit and UI tests.

Coverage report for jenkins has been generated through the django\_jenkins plugin. The django\_jenkins plugin generates coverage.xml report file which can be read by the Cobertura plugin on Jenkins. Additionally a Junit.xml test report file is generated which enables Jenkins to display the test report and graph of test successes from previous builds.

Statical code analysis has been performed through Pylint. Pylint has been configured to ignore warnings that are erroneously generated on Django files. Summary of Pylint analysis has been submitted to the team folder. Summary of Pylint report

[Pylint report](https://drive.google.com/a/bu.edu/folderview?id=0B7fWDHo1Z7OvWDVVWE5kTmZJWWc&usp=sharing_eid&ts=569ea45b&tid=0B7fWDHo1Z7OvNkw1ZXhwd0xscHc)

Total score for our code based on Pylint report : 6.5

**Process Metrics**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Iteration | deadline | Sub group | Task detail | | | Estimated total time | Actual total time |
| #of Task | Estimated hour | Actual hour |
| Iteration I | 2016/3/2 | Nora & zhi | 2 | 5 | 5 | 3 weeks | 3 weeks |
| Div & Abhyu | 3 | 10 | 8 |
| Minteng & Rui | 2 | 5 | 5 |
| Chencheng & Ruirong | 4 | 8 | 4 |
| Iteration II | 2016/3/29 | Nora & zhi | 3 | 12 | 16 | 3 weeks | 4 weeks |
| Div & Abhyu | 2 | 12 | 11 |
| Minteng & Rui | 2 | 12 | 10 |
| Chencheng & Ruirong | 2 | 12 | 8 |
| Iteration III | 2016/4/27 | Nora & zhi | 3 | 12 | 18 | 3 weeks | 4 weeks |
| Div & Abhyu | 2 | 16 | 12 |
| Minteng & Rui | 1 | 12 | 12 |
| Chencheng & Ruirong | 1 | 24 | 20 |

## Standard

**Coding Standard:**

We will follow the PEP 008 Style Guide for Python Code. Details about this coding standard is available in the link below:

https://www.python.org/dev/peps/pep-0008/#code-lay-out

We have used the Pylint checker to check for Python coding standards. The coding standard violations on some of the files has been fixed. Most coding standard violation relate to line length and variable names. Jenkins has been set up along with pylint to check for coding standard violations. Future commits could be checked against previous commits for coding standard violation which will ease up the code review process.

**Documentation Standard:**

Documentation to be added to Github.

Comments should be added along with the code in the source files.

## Inspection/Review Process

Each commit should be subject to review before being placed in the staging area.

Each commit was reported during the team meeting. The Implementation leader reviewed the diff before the file was committed to branch. A fresh branch was maintained in a clean state at the beginning of the project. Every individual team committed to their own branch and the code was integrated to the iteration branch after review by the Implementation leader. As part of the code review process, test cases have been checked.

## Testing

i. **Unit Testing** Scripts using unittest (PyUnit) Module. Unit test scripts will be run on each build to update Reporting by each team member.

ii. **User Acceptance testing(UAT)**: at the end , we will test if all the requirement proposed initially are met which will be tested by real unbiased audience in real environment.

## Defect Management

Criteria of defect : Defects are any error that results in failure to conform to specification or failure to operate properly or function efficiently.

Defining defects involves four specification: category it belongs to, severity, priority and status.

1. **Defect category type:**
   1. **Software engineering process defect :**

Design flaw:

* Icebox placement
* Story movement iteration to icebox
* No iteration
* No role when sign up
* No constraint on iteration date
* duplicate project name
* 0 hours for user story

Test flaw:

* unit/test\_iteration is blank (no test in this file)
* unit/test\_other has error
* unit/test\_project\_api (include iteration test, which should refactor to the ‘test\_iteration’)
  1. **Configuration defect :** defect involving software patch flaw or upgrade flaw of tool used in project which are installed , administered, compile or resourced or misconfigured in a manner resulting in improper system operation, introduce vulnerabilities and degrade system performance.

The Current environment requires to deploy a virtual machine to work on while by using python’s virtual environment package. We can just simply work on local environments. Further, the best the development of this project is to deploy a virtual machine on cloud at the very beginning.

* 1. **Code Defect: (**flaws in the code implemented**)**:

In previous test code, there are tons of flaws: 1)no users was created before running the test. 2)data and function are integrated with each other. 3) they assigned a fixed username and password to run all the tests, which means you have to know the username and password at first. Further, in some specific test case, they just assign ‘Bob’ or ‘George’ in many UI test and it is not reusable. 4)In all UI tests, the use of selenium’s function like ‘find\_element\_by\_xpath’ is not reasonable and reuseable. Instead of counting which ‘div’ it is, test should use the id of the ‘div’ when trying to click the specific button in case of any modification in the template in the future.

1. **Defect Severity :** Defect severity is classified according to their impact: fatal(blocks the current task), major(flaw in prime functionality of system causing system failure), minor(not causing system failure but flaws in functionality of system) and cosmetic(which does not hamper the any functionality of system).

To determine severity we ask ourself few question regarding the defect such as is it blocking the system functioning? is system able to recover on its own ? Answers to these question determines the level of severity.

All defects are considered minor, since they do not cause system failure.

1. **Defect priority :** It’s different from severity and not related to it. Priority is relative importance of addressing and resolving the defect. levels are 1. critical , 2. essential and 3. Cosmetic.

Critical:

* No constraint on iteration date
* duplicate project name
* 0 hours for user story

Essential:

* Story movement iteration to icebox
* No role when sign up

Cosmetic:

* Icebox placement
* No iteration

1. **Status:** the current status of the state of defect (open, closed or reopened)

All defects are solved & closed.

**Defect Management tool for reporting :**

1. GitHub

Regressions and Deployment Bugs

2. Pivotal Tracker

New requirements that arise as a result of bug or issue

3. Slack

To get noticed by the individual for the defects encountered by him/her during unit testing. Discuss ideas and reach out to team members on any problems for help.

4. Jenkins

Run all the test, help configuration and generate the report.

# Configuration Management Plan

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

## Configuration items and tools

* + 1. Git/Github
    2. Pivotal Tracker
    3. Google Docs

## Change management and branch management

* + 1. Branch Management
       1. master: The original release branch.
       2. Work: branch for merging
       3. Subgroup branches: group1, group2, group3, group4
       4. Final: the branch for the final version
    2. Change management

we change the code, because there are bugs or we want to add new feature.

* + - 1. in each iteration, every one should understand the goal of changing this code in this iteration.
      2. Under the same goal, different subgroups should have different work or tasks.
      3. If someone wants to add a new feature or found a new bug, he should report to the whole team and after approval from the whole team, this feature or bug could be add to the work.

## Code commit guidelines

* + 1. you should commit code only if someone else’s work depends on the feature you make otherwise everyone could and should commit the code at the end of each iteration.
    2. Before committing, your code should be guaranteed by your partner, and get approval of all team members.
    3. In each commit, the remark title should be new feature “feature tile” or bug fix “bug-fix title”, and with author name.
    4. The code you submit should be well commented.

# 

# Security Process and technique

a. Security Training

I. Common Security Issues in software development

Search and know some common security issues in software development, and then avoid those issues as far as possible.

II. Some good links and readings

Chapter 20: security in “The django book”, “Security in django”, we share those in our team folder.

b. Security Requirement

I. Security Requirement Definition

A user-oriented quality requirement that specifies a minimum required amount of a quality sub factor of the security quality factor.

II. User Requirement

The security requirement depends on the user requirement, developers have to find what the user need in the security aspects.

c. Security Design

I. Data Secrecy

II. Data Integrity (Accuracy, Reliability)

Developers need to ensure the accuracy and reliability of the data in the software development.

III. System Availability

We firstly need to fix the bugs of the existing software; we don’t want to destroy the whole system. So we need to ensure the availability of the system. We use github to implement version control to maintain system availability.

IV. Exception Handling (Error Handling)

d. Secure Coding

I. Maintain existing code

Read and learn the existing code, also fix bugs and maintain the system by secure coding.

II. Secure coding

There are many secure coding standards, guidelines and publications on the Internet. We share those links in our team folder.

e. Security Detection/testing

For security detection, we find “OWASP Python Security Project”, and we also use pylint.

I. Static Detection (security in python)

White-box analysis, structural and functional analysis

II. Dynamic Detection (security of python)

Black-box analysis, identify and address security-related issues.

III. Exception Detection (security with python)

Detect the error and exception handling of the software. Develop security-hardened python suitable for high-risk and high-security environments.

f. Security Supervision

I. Risk Assessment

The security supervision and risk assessment need to be considered in every process of the software development. We find django-supervisor.

# References

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

* http://www.amsaa.army.mil/Images/RiskReportingMatrix.png
* <http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf>
* http://blog.capterra.com/agile-project-management-software/
* http://reviews.financesonline.com/p/pivotal-tracker/#benefits
* https://app.activecollab.com/119544/people
* http://project-management.softwareinsider.com/l/219/Active-Collab
* <https://www.quora.com/What-are-activeCollabs-pros-and-cons-Are-there-better-alternatives>
* <https://en.wikipedia.org/wiki/Jenkins_(software)>
* <http://searchsoftwarequality.techtarget.com/definition/Jenkins>