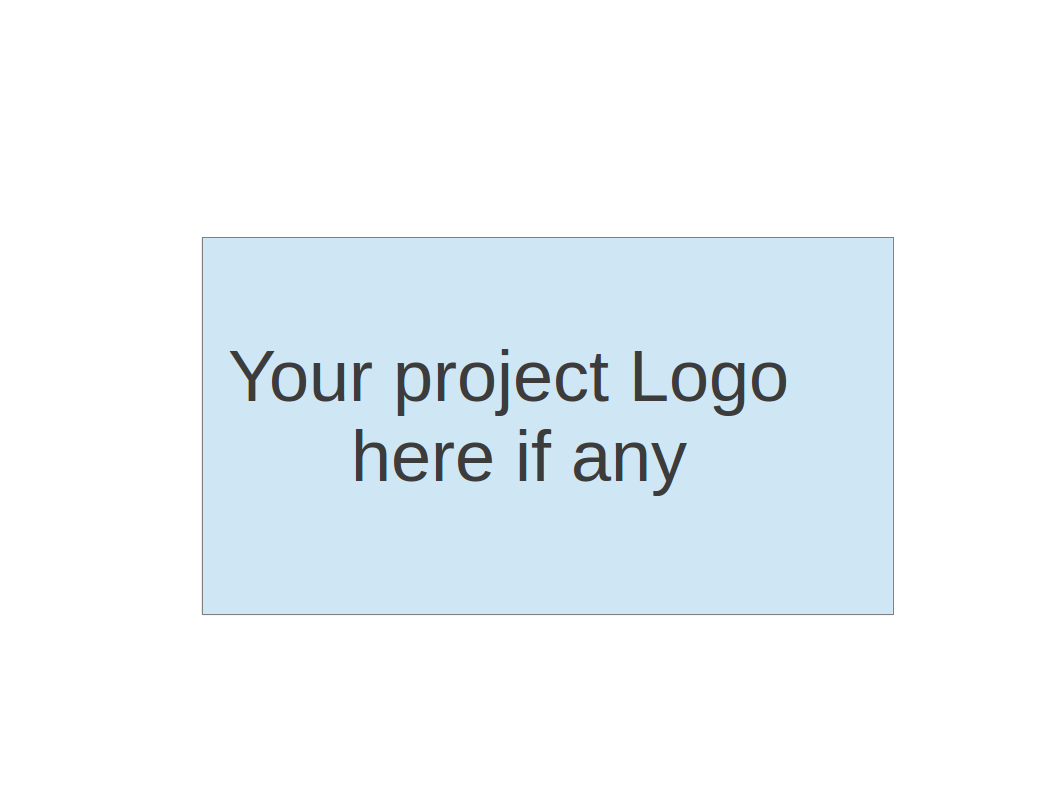
**CS673F14 Software Engineering** 

**Group Project - Online Project Planner**

**Project Proposal and Planning**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Daniel Abramowitz | Co-Design Leader | *DA* | 09/21/2014 |
| Daniel Carlos | Configuration Leader |  |  |
| Ray Gborsongu | Co-Design Leader and Co-Quality Assurance Leader | *GR* | 9/21/2014 |
| Luis Marion | Project Leader | LM | 09/24 |
| Nethra Balasundaram | Back-up project leader /  Co-Requirements Leader | *NB* | 09/19/2014 |
| Mariana Arantes | Co-Requirements Leader | *MA* | 09/21/2014 |
| Roberto Reyes |  | *RR* | 09/24/2014 |
| Olufemi Odegbile | QA leader | *OO* | 09/24/2014 |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
|  | Luis Marion | 09/18/2014 | adding role |
|  | Daniel Abramowitz | 09/18/2014 | adding role |
|  | Luis Marion | 09/24/2014 | adding monitoring/controls and project schedule sections |
|  | Daniel Carlos | 09/24/2014 | adding Configuration Management Plan |
|  | Olufemi Odegbile | 09/24/2014 | Adding metrics and standard |
|  | Mariana and Nethra | 09/24/2014 | Overview, Related Work, Detailed Description |
|  | Dan and Ray | 09/24/2014 | Process Models, Risk Management, Objectives and priorities |
|  | Ray | 09/24/2014 | Testing |
|  | Roberto | 09/24/2014 | Defect Management |
|  | Luis Marion | 10/02/2014 | Revised the schedule |

[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. )

ProManage is a free,web based Project Management tool largely inspired by PivotalTracker, provides a complete, online solution for managing time, deadlines, work tasks, teams and people - all in a secure online project management tool. Designed to adapt to an organization’s business processes, ProManage delivers a highly flexible and configurable portfolio for a wide range of project teams and project types.

# **Related Work**

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

Several project management tools are available including both desktop and online tools. Few famous tools are MS-project, Basecamp, Trello and PivotalTracker. MS-Project is a desktop version of a project management tool, which helps project manager to keep the project flow on track. BaseCamp, Trello and PivotalTracker are online web-based project management tools, allows the user to invite as many members as they want for their project for free or with monthly charges. Each one differs in terms of ease of access, UI design and process model.

Our product gives easy to use UI, follows agile process model, due based management rather than priorities, allows communication thread for discussion on every task and is targeted for users ranging from students to real-time project managers. Our product is highly efficient for small to medium projects.

# **Detailed Description**

(include some preliminary functional and nonfunctional requirements )

**3.1 Functional Requirements**

The Project Management Tool is a web based application and the access is permitted only for registered users. The users will be able to:

* Create a new account in the system, using the sign up feature;
* Log in the system, using the username and password created in the sign up step;
* Log out of the system;
* Retrieve password;
* Delete account;

The purpose of the system is help users to manage their projects and, because of that, after the user is logged, the follow activities are possible:

* Create a project (A project has name, description and members);
* Invite users to the project (only registered users);
* Assign role to members (the project can have one Project Leader and the other members are Team Members);
* Remove members from project;
* Delete project;

When users log in the system, they are redirected to their home page, where they are able to see all the projects they have. When the user selects a project, the home page for the project is open. The activities allowed are:

* Create task (A task contains name, description, due date, assigned members, testing strategy and estimated time);
* Assign users to tasks (A user can be assigned to many tasks and one task can have many users);
* Search tasks (Tasks can be searched by due date, creation date, user assigned, finished date);
* Update task ;
* Mark task as done;
* Delete task;

A project can have many milestones and the Project Leader will be able to:

* Create milestones (A milestone has a name, description and date);
* Update milestones;
* Delete milestones;

Managing time is a big challenge in most of the projects and to help our users to keep their schedule and finish their tasks on time, the system will have:

* A feed to inform users about every change in the project (tasks created, tasks updated, tasks finished and changes in the team);
* A calendar;

The system will also send by email, to the members of the team, all the changes made in the system and reminders of tasks that are approaching their due date.

**3.2 Non Functional Requirements**

There are a lot of different users that would like to use the Project Management Tool and to offer a good service, reaching the most number of them, the requirements are:

* The system will work properly in the most popular web browsers: Mozilla Firefox, Google Chrome, Safari and Opera;
* The system will be always available (In case of maintenance, it will be made at night on weekends);
* The system will provide access control mechanism through the use of password and user ID. This will ensure that only authorized team members can access the tool. The system will distinguish between member and manager roles and apply permissions accordingly.

# **Management Plan**

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

## Process Model

## Our group project management project type is an iterative process, specifically, an agile project management tool for managing software development projects.

## The imposed process model for development is implicitly agile and is characterized by short weekly meetings to discuss requirements, design, implementation and testing.

* We will require a minimal working prototype while adding additional features, as required by the customer.

## Objectives and Priorities

## Project objectives are defined over time, cost and quality.

* Assess similar projects and improve upon their design.

## The expected completion time is first week in December 2014 at a minimal cost of disruption.

## The team is resolved to deliver a quality product that closely meets functional and nonfunctional needs of stakeholders which include security,portability, maintainability and availability.

## The team priorities are setting up and configuring development tools, agreeing on the type and scope of project management to implement, creating user stories and corresponding test cases on pivotal tracker during the first iteration and producing the necessary documentation at the planning phase of the project.

## Risk Management:

## During each iterative phase, risks that threatens implementation of a user requirement and the development process will be determined and assessed in terms of impact and probability of occurrence.

## Appropriate risk mitigation methodology will be applied to reduce significant risks to acceptable levels.

## Monitoring and Controlling Mechanism

* + 1. Group will meet at least once weekly - Sunday 8 pm - 9 pm via skype.
       1. Group will determine if additional meetings/in-person meeting are required for subsequent week at each meeting.
       2. Minutes are tracked in shared document on google-docs.
       3. Group will (optionally) meet before class as team member schedules permit.
       4. Weekend meeting topic will be established in at the end of class.
    2. All team documents are shared on google-docs and will be shared with professor. Documents will be added to github as required by the professor.
    3. Github will be the source control system of choice. Code reviews will take place in order to enforce standards and sign-off on code.
    4. Pivotal Tracker will be used in order to keep track of user stories and project tasks and ensure deadlines are being met.

## Schedule and deadlines

|  |  |  |  |
| --- | --- | --- | --- |
| Dates | Week | Title | Description |
| 09/04/2014 | 1 | First Day of Class | Introductions and team forms. |
| 09/04/2014-09/11/2014 | 2 | Team Norms | Meet to establish team norms, roles, meeting schedule and term project subject. |
| 09/11/2014-09/18/2014 | 3 | Market Research | Compare and contrast different vendor products, including previous teams term project. |
| 09/18/2014-09/25/2014 | 4 | Project Proposal | Create user preliminary user stories for the project. Create a couple of user stories for client project. Complete SPPP document. Create presentation for Project Proposal. |
| 09/25/2014-10/02/2014 | 5 | Personal Project Tracker | Create the base data model. Data model should include core functionality (login, stories, tasks, etc.) |
| 10/02/2014-10/07/2014 | 6 | Personal Project Tracker | Begin the following user stories:  User registration  Project Creation  Story Creation  Task Creation  Across all application tiers (ui, service, database) |
| 10/07/2014-10/16/2014 | 7 | Personal Project Tracker | Compete outstanding work on stories and test them. |
| 10/16/2014-10/23/2014 | 8 | Group Project Tracker | **Iteration 1 is complete. Present.**  Begin following user stories:  Invite users to projects.  Assign/update stories, tasks, projects  Authorization based on user roles  Across all app tiers (ui, service, db) |
| 10/23/2014-10/30/2014 | 9 | Group Project Tracker | Finish update/assignment coding. Finish authorization code to services to enforce permissions. |
| 10/30/2014-11/06/2014 | 10 | Group Project Tracker | Test group features and regression test the “personal” features of the tool. |
| 11/06/2014-11/13/2014 | 11 | Polish | **Iteration 2 is complete. Present.**  Revisit the ui’s and make sure they appear consistent stylistically across the screens. Evaluate if any optional features can be added in time remaining. |
| 11/13/2014-11/20/2014 | 12 | Polish | Complete outstanding ui work and any additional optional features (and/or pull them). |
| 11/20/2014-11/26/2014 | 13 | Acceptance Testing | All functionality is completed. User testing taking place. Defects which are high priority or low complexity may be addressed. |
| 11/27/2014 | N/A | Thanksgiving Day | Enjoy the Holiday! |
| 11/28/2014-12/04/2014 | 14 | Presentation | Focus is on creating presentation for the project and fixing any remaining (low risk) defects. |
| 12/04/2014 |  | **DEADLINE** | **Iteration 3 is complete. Present final project. Good luck on final exam.** |

# 

# **Quality Assurance Plan**

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

## Metrics

We are going to categorize metrics for this software project into two: directly measured and indirectly measured metrics.

Direct measurements (internal attributes) are: Cost(measured in hours invested in developing the software), LOC (line of codes written), KLOC (kilo line of codes), speed, size of the source codes in bytes.

Indirect measurements (external attributes) are:

* Quality: We will measure this attribute by measuring
  + Correctness: degree to which the the program operates according to specification. This will be achieved by using the following derived metrics: defect/KLOC and failures/(hours of operations). Our goal for this project is to minimize the two metric to zero.
  + MTTF(mean time to fail). Failure in this case means when program crashes or freezes. We will measure the time in calendar or processor time. Our goal is to make sure that MTTF is as large as possible.
* Integrity: Measures degree to which the software withstand outside attack such as unauthorized accesses. We will require that a user input a strong password to create an account.
* Maintainability: Measures degree to which the software can be modified: add-ons, patches, error-corrections, extensions. We will use cost to make correction as a metric to this attribute. Our goal is to make the cost to make correction reasonable.
* Complexity: We will use LOC and McCabe's complexity measure as metrics to measure complexity of our software. McCabe's metric is based on a control flow representation of the program. We will try to write the code such as the McCabe’s metric is less than 10.
* Usability: This is the measurement of easiness of use. The metric we are going to use is *training time* for new users and *skill level needed to use the software*.

## Standard (e.g. documentation standard, coding standard etc. )

We will use Google Doc to write our documentation. We will partly follow ISO/IEC 26514: 2008, *Systems and software engineering – Requirements for designers and developers of user documentation*. Our software’s user documentation will have the following components: Identification data, Table of Content, Lists of Illustrations, Introduction, Information for use of the documentation, Concept of Operations, Procedures, Information on Software Commands, Error Messages and Problem Resolution, Glossary, Related Information Sources, Navigational Features, Appendix. The writing responsibility will be shared among the group members. Any additions/modifications to the documents by a group member will be in red letters for the group to review and edit. Drafting, checking, revising and re-drafting is an iterative process that will be continued until a document of acceptable quality is produced.

We will follow the *CERT Oracle Secure Coding Standard for Java* created by Joe McManus when coding the software using java.

## Inspection/Review Process

(e.g. describe what are subject to review, when to conduct review, who do the reviews and how ?)

1. Documentation is subject to review one week before major deadlines (iteration 1,2, and 3). Discrepancies will be detailed by Quality Assurance Leader and corrected by other team members.
2. Code is subject to review at varying times. Each team member should be accountable for any work that they produce. this kind of inspection should occur before any piece of code is pushed to GitHub. The code will be inspected by the writer for obvious coding bugs.  
     
   Additionally, before the deadlines for each iteration, code must be reviewed by other team members. This effort will be coordinated by the QA leader.
3. Baseline Quality:  
   Before the end of an iteration, the baseline quality will be reviewed. The QA Leader will divide the task of determining whether or not the software meets functional requirements. Furthermore, any changes to the module design should be identified, modules should be tested, and the functionality or lack thereof should be documented.

## Testing (e.g. who, when and what type of testing to be performed? How to keep track of testing results?)

## The teams project will be built on a three tier architecture which includes client tier, business tier, and the storage / persistence tier. It is expected that testing will cover each tier.

## At the client tier , user interface elements such as input and output forms, dialog boxes, menu items and others will be tested for ease of use. These interfaces will also be tested to conform to design specifications and should function according to user requirements.

## At the business tier, business rules or logic that process client request from the client tier will be tested for functionality, speed and quality of response.

## At the storage tier, the database elements such as entity or object names should be meaningful and descriptive; data, relationship and key constraints should be verified and validated; procedures, triggers, functions or query results should be accurate, complete and timely.

## The type of tests include unit, module, integration, regression, sociability and user acceptance test.

* Requirements traceability matrix will be used to track testing results.

## Defect Management

(e.g. describe the criteria of defect, also in terms of severity, extend, priority, etc. The tool used to management defect, actions or personnel for defect management)

Defect management will be conducted with a document on GitHub logging the date of the defect discovery, the user who discovered the defect, and the location of the defect (directory path), severity, priority.

It may not be practical to eradicate all defects, but 80% of all discovered defects should be dealt with.

1. Criteria
   1. The criteria for a defect includes a function’s deviation from its expected functionality. If the task is completely unimplemented, this is a serious defect. If the task is completed but occasionally does not work, the defect is moderate. Low priority defects include defects such as typos and aesthetic issues.
2. Priority
   1. Major defects must be dealt with before the end of the iteration.
   2. Most Moderate defects must be dealt with by the end of the iteration, but some (~10%) may be tolerated.
   3. Many minor defects can be tolerated (80%).

# **Configuration Management Plan**

**6.1 Introduction**

The Software Configuration Management Plan (SCMP) describes how the artifacts for the Pro-Manager are to be managed.

**6.1.1 Acronyms**

CI: Configuration Item - an item tracked by the configuration system.

CM: Configuration Management - the process of maintaining the relevant versions of the project.

SCMP: the Software Configuration Management Plan (This part of the document [6] )

**6.2 SCM Management**

**6.2.1 Organization**

One person will be designated as the “configuration leader” for the duration of this project.

**6.2.2 SCM Responsibilities**

Configuration Leader: Shall be responsible for organizing and managing configuration management (CM), maintain the SCMP, install configuration management tool(s).

Project Leader: Will take over the configuration leader’s function only under exceptional circumstances.

Engineers: Accept the CM rules that configuration leader choose.

**6.3 Applicable Policies, Directives and Procedures**

All current and previously released versions of CIs will be retained;

The Pro-Manager will use git, a control version system, with GitHub (<https://github.com/>) for the control version of the source code.

The Pro-Manager will use Google Docs (https://docs.google.com), by Google, for the control version of the project management documents.

**6.4 Configuration Identification**

The configuration leader shall be responsible for identifying all CIs. The team leader shall approve them.

**6.5 Configuration Control**

After a group of CIs are tagged, a baseline is defined. To request a change in a baseline the member shall email all the whole team and after a discussion the configuration leader shall be responsible for approving or disapproving changes.

**6.6 Version Control System - Source Code**

**6.6.1 Code commit guidelines**

A commit should not have unrelated changes. A commit should be easy to review and merge.

Changes should never be committed before properly tested.

Write your commit in the present tense: “Fix bug” and not “Fixed bug.” This convention matches up with commit messages generated by commands like git merge and git revert.

Write a title and a summary to better explain changes.

Write the PivotalTracker story ID in the commit message.

**6.6.2 Branch management**

There are two main branches with infinite lifetime: master and develop.

The main branch always reflects a production-ready state.

The develop branch always reflects a state with the latest delivered development changes for the next release. When the source code in the develop branch reaches a stable point and is ready to be released, all of the changes should be merged back into mastare somehow and then tagged with a release number.

There are also features branches. Those are used to develop new features for the upcoming or a distant future release. May branch off from develop; must merge back into develop; the branch naming convention is anything except master, develop.

# References (do we need this?)

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

# Glossary