ProManage

Luis Marion
Mariana Arantes
Roberto Reyes
Daniel Carlos
Nethra Balasundaram
Daniel Abramowitz
Ray Gborsongu
Olufemi Odegbile

Contents

Requirements Analysis

This 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
- Log in to the system
- Log out of the system
- Delete account

Functional Requirements

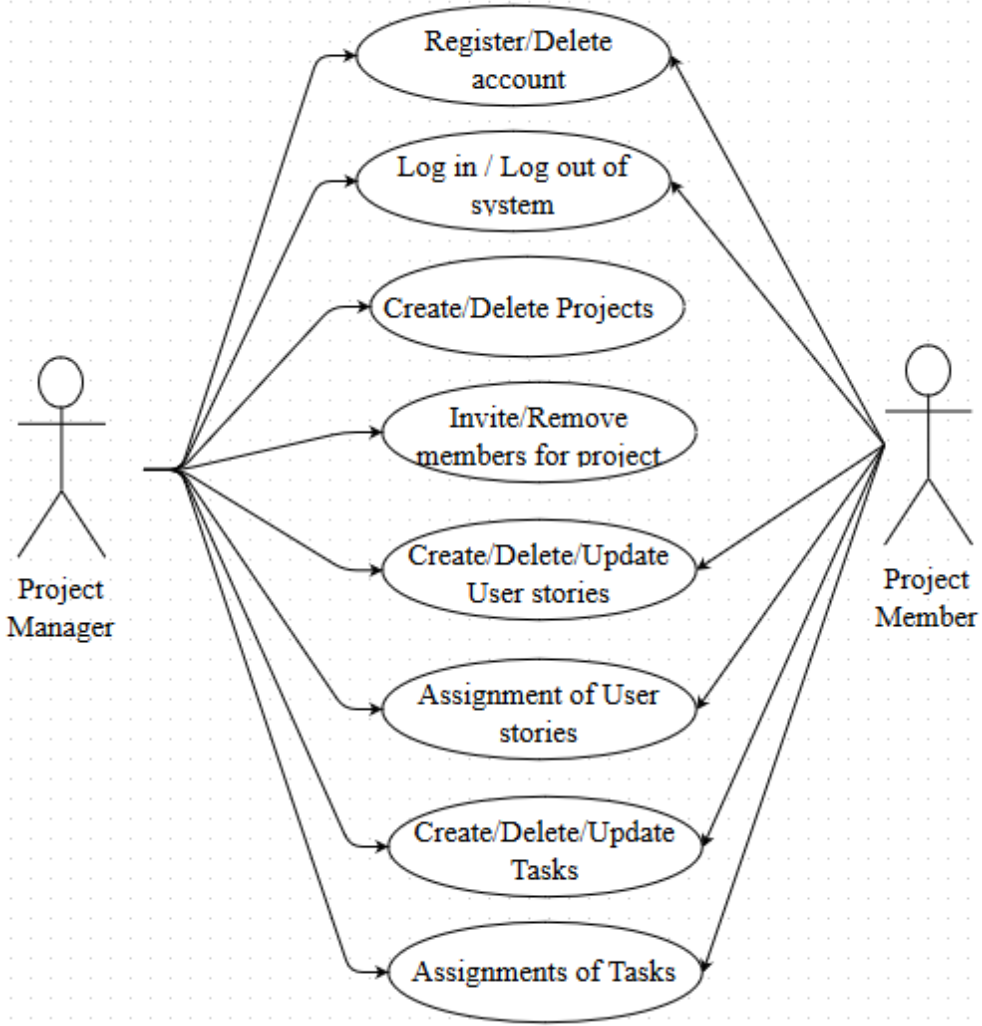
The purpose of the system is help users to manage their projects and, the following activities are possible:

- Create/Edit/Close a project
- Invite users to the project
- Assign role to members
- Remove members
- Create/Delete stories
- Create task
- Assign users to tasks
- Update task
- Mark task as done
- Delete task

User Stories - Sample

- **Create Users**
  - As a user, I want to be able to register, so that I am able to use ProManage.
- **Create Projects**
  - As a registered user, I want to be able to create and delete a project in ProManage, so that I can take advantage of the application to manage my project.
- **Create Stories**
  - As project team member, I want to be able to create a users stories for the project, so that I can assign jobs to members of project.
- **Create Tasks**
  - As a project team member, I want to be able to create task specific to the project, so that I have detailed task for each story.

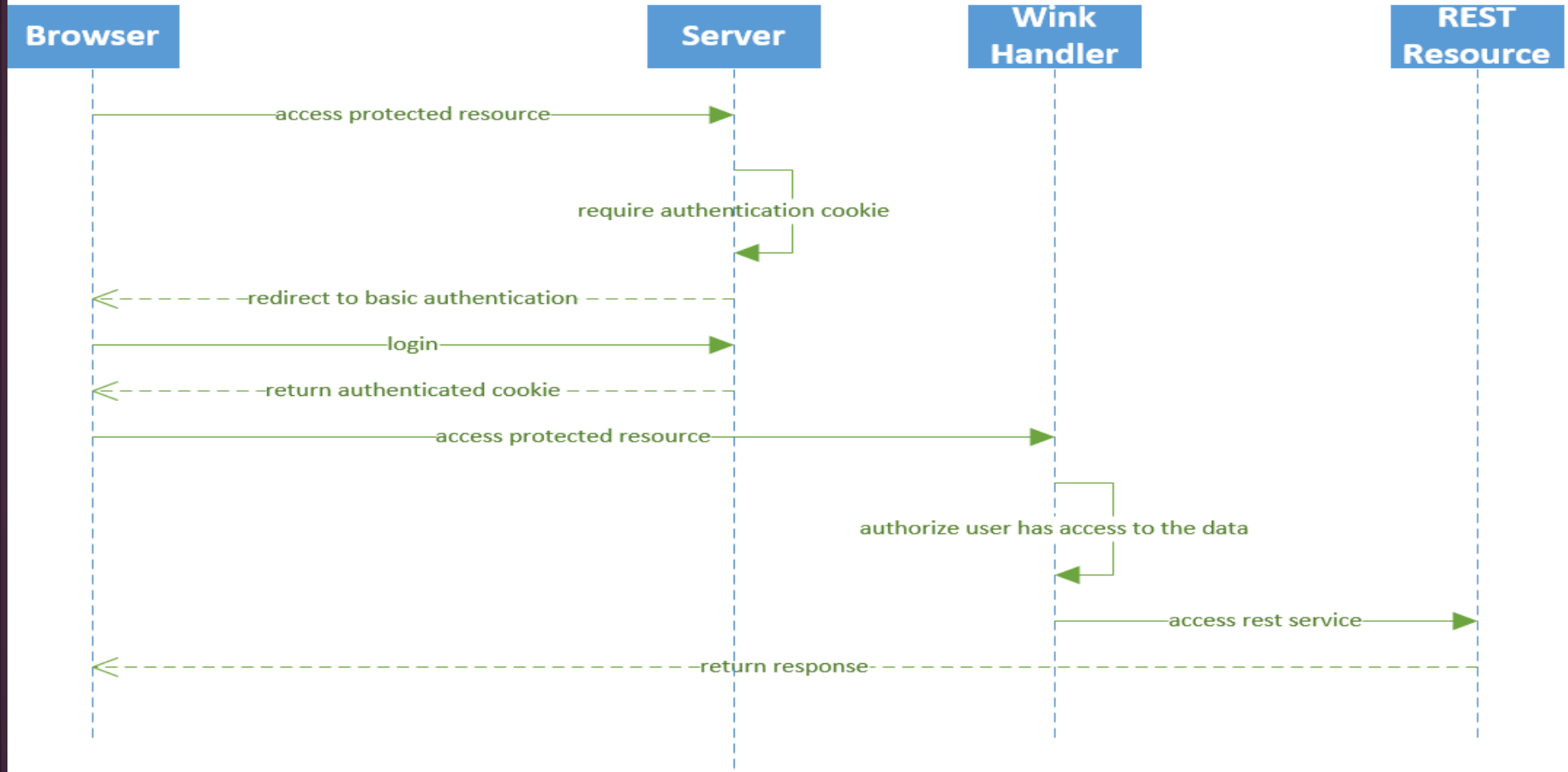
Use Case Diagram



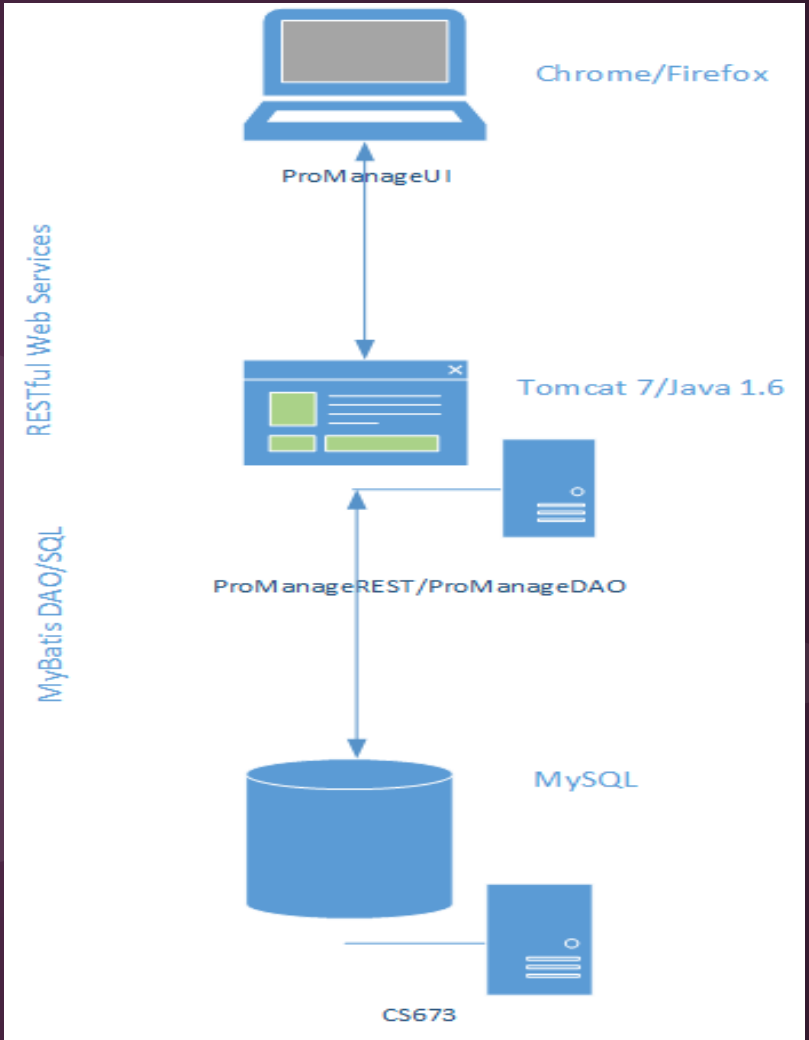
Non Functional Requirements

Maintainability	➔ extended with new functionality ➔ code review, refactoring and optimization
Availability	➔ incident management- recovery from crashes, failures ➔ speed of providing services - response times
Supportability	➔ Support users with problems
Security	➔ access control - user account provisioning (access rights and privileges), identification, authentication and authorization ➔ segregation of duty - assignment of stories, task, projects according to some criteria
Portability	➔ Supports many web browsers including firefox, safari and internet explorer

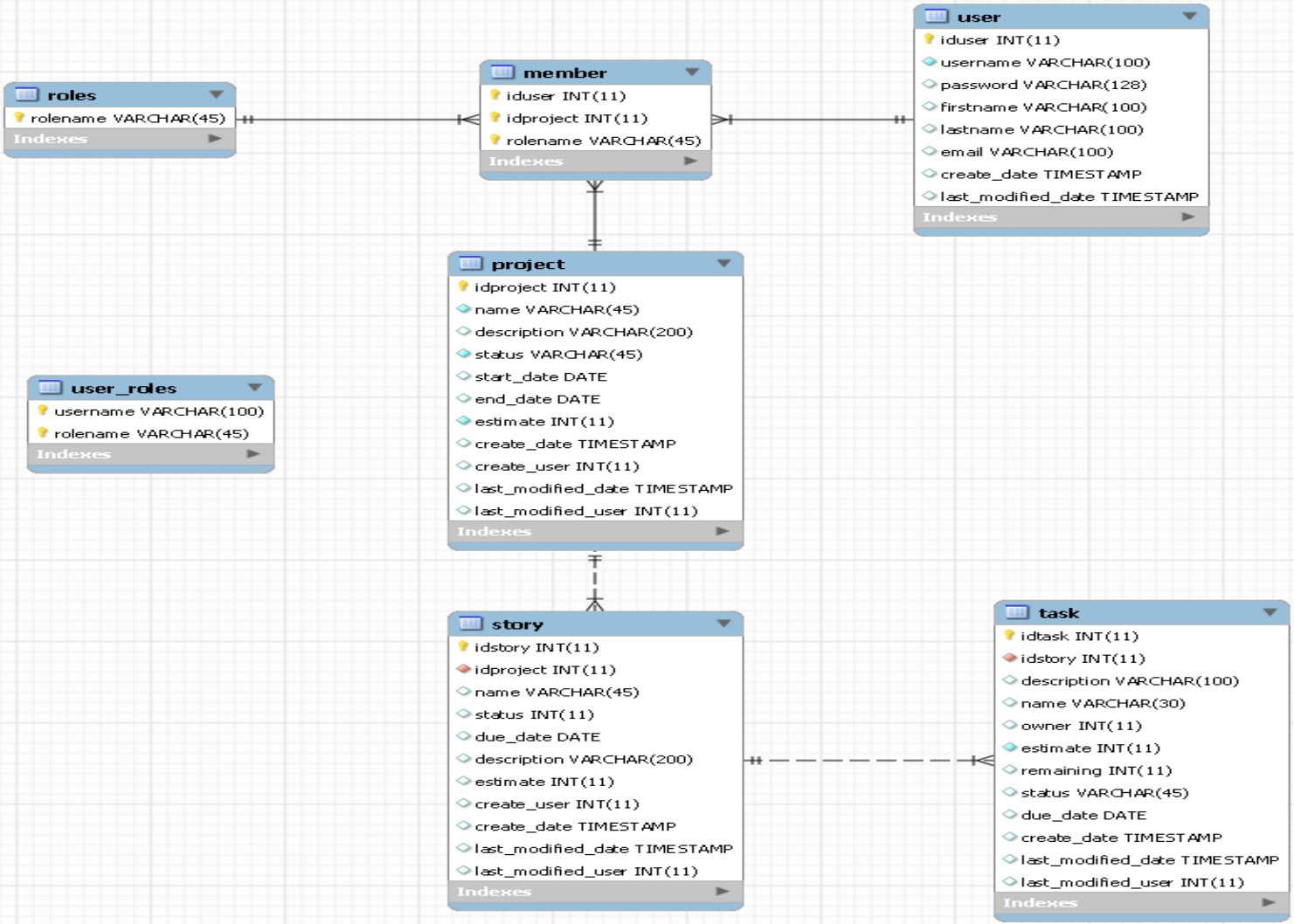
Non Functional Requirement - Implementation Authentication/Authorization



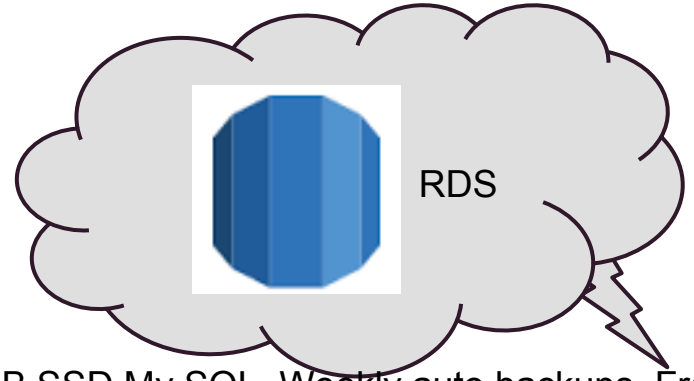
Design Architecture



Design -Data Model



(AWS)-Amazon Web Services



5GB SSD My SQL. Weekly auto backups. Free.
Located in Oregon (slow)

Linux virtual machine running apache 7.0 server. (free is also located in Oregon)

Implementation- Object Generation

Entity Objects or Model (Persistent Objects)	**Control Objects (Action UI Elements)**	**Boundary Objects or View (Interface elements)**
User	Sign UP/IN/OUT/, Delete, Invite, Credential Recovery	Pages for signing up, signing out, signing in, to search user by username and to set security credentials
Project	Create, Retrieve/List, Update, Close	A page to list projects, to show project details and manage project
Stories	Create, Retrieve/List, Update, Delete, Close	A page for managing User Stories
Task	Create, Retrieve/List, Update, Delete, Finish	A page for managing Task
Roles	updateRoles,	A page to add user to project as a member.

Implementation - UI

- Bootstrap framework for html and css
  - helps to create responsive webpages
  - easy to learn and use
- AngularJs framework for html and js
  - lets the user extend html vocabulary
  - it is expressive, simple and clean

Implementation - UI

Use of ng-view to change pages and share data between them

```
app.config(function($routeProvider) {
    $routeProvider
    .when('/project/:projectId', {
        templateUrl : 'projectview.html',
        controller : 'ProjectViewCtrl'
    }).when('/newproject/:projectId', {
        templateUrl : 'newproject.html',
        controller : 'FormController'
    }).when('/members/:projectId', {
        templateUrl : 'members.html',
        controller : 'MembersController'
    }).when('/addmember/:projectId', {
        templateUrl : 'addmember.html',
        controller : 'AddMemberCtrl'
    }).when('/newstory/:projectId/:storyId', {
        templateUrl : 'modalNewStory.html',
        controller : 'NewStoryCtrl'
    }).when('/errormsg', {
        templateUrl : 'errorMsg.html',
    }).otherwise({
        templateUrl : 'projectList.html',
        controller : 'ProjectsController'
    })
})
```

Implementation - Refactoring

Dashboards | Projects | Measures | Issues | Rules | Quality Profiles | Quality Gates | Log in | Search

Promanage DAO project analyzed with the SonarQube Runner

Dashboard
Issues
Time Machine

TOOLS
Components
Issues Drilldown
Design
Libraries
Compare

sonarqube

Version 1.0 - Quarta, 03 de Dezembro de 2014, 19:02 h | Time changes...

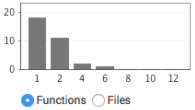
Lines Of Code	Files	Functions	SQALE Rating	Technical Debt Ratio
830	12	32	A	0,9%
Java	Directories: 2	Lines: 1.140		
		Classes: 12		
		Statements: 319		
		Accessors: 79		

Technical Debt	Issues	
3h 42min	117	Blocker 0
		Critical 1
		Major 103
		Minor 13
		Info 0

Duplications
0,0%

Lines	Blocks	Files
0	0	0

Complexity
1,8 /function
4,7 /class
4,7 /file
Total: 56



Events | All

03/12/2014	Version	1.0

Promanage DAO project analyzed with the SonarQube Runner
java-sonar-promanageDAO
Profiles: Sonar way (Java)

SonarQube™ technology is powered by SonarSource SA
Version 4.5.1 - LGPL v3 - Community - Documentation - Get Support - Plugins - Web Service API

Embedded database should be used for evaluation purpose only

localhost:9000/dashboard/index/4

Implementation - Refactoring

sonarqube

Rule	
Left curly braces should be located at the end of lines of code	76
Right curly brace and next "else", "catch" and "finally" keywords should be located on the same line	18
System.out and System.err should not be used as loggers	3
Class variable fields should not have public accessibility	1
if/else/for/while/do statements should always use curly braces	1
Exception handlers should preserve the original exception	1

Implementation - Refactoring

sonarqube

```
public class ProjectDAO
{
```

Move this left curly brace to the end of previous line of code.

Open | Debt: 1min

```
    public static String PROJECT_LEADER_ROLE = "project_leader";
```

Make PROJECT_LEADER_ROLE a static final constant or non-public and provide accessors if needed.

Open | Debt: 10min

Test Case

Test Case	Expected Result	Status (Pass, Fail, Blocked)
Enter a valid user name and a valid password, click login button	Application should display home page	Pass
Enter a valid user name and invalid password, click login	Application should display an error message and re-open the login page	Pass
Log out, enter an invalid user name and valid password, click login	Application should display an error message and re-open the login page	Pass
Log out, enter an invalid user name and invalid password, click login	Application should display an error message and re-open the login page	Pass

Testing

- MemberDAOTest.java
- ProjectDAOTest.java
- StoryDAOtest.java
- TaskDAOTest.java
- UserDAOTest.java

```java
public class UserDAOTest extends TestCase {

    @Test
    public void testGetProject()
    {
        UserDAO dao = new UserDAO();
        User user = dao.getUserByName("cyclops");
        assertNotNull(user);
    }

    @Test
    public void testGetUserByProject(){
        UserDAO dao = new UserDAO();
        List<User> userByProject = dao.getUserByProject(1);
        for (User user : userByProject) {
            System.out.println(user.getUserId());
        }
        assertNotNull(userByProject);
    }
```

Testing

```
public User getUserByName(String username) {
    User user = null;

    SqlSessionFactory factory = SessionFactorySingleton.getInstance()
            .getSqlSessionFactory();
    SqlSession session = factory.openSession();
    try {
        user = (User) session.selectOne("selectByName", username);

    } finally {
        session.close();
    }
    return user;
}
```

Green = Fully covered, Red = not covered, Yellow = partly covered

Testing - Iteration 2

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
ProManageDAO	23.1 %	474	1,580	2,054
src	29.4 %	356	856	1,212
bu.met.cs.cs673.pm.dao	28.0 %	231	595	826
UserDAO.java	19.1 %	42	178	220
StoryDAO.java	14.9 %	25	143	168
ProjectDAO.java	29.7 %	51	121	172
TaskDAO.java	32.9 %	49	100	149
MemberDAO.java	39.8 %	33	50	83
SessionFactorySingleton.java	91.2 %	31	3	34
bu.met.cs.cs673.pm.dto	32.4 %	125	261	386
User.java	0.0 %	0	76	76
Story.java	28.8 %	30	74	104
Task.java	44.8 %	39	48	87
Project.java	43.8 %	35	45	80
Member.java	58.3 %	21	15	36
Role.java	0.0 %	0	3	3
junit	14.0 %	118	724	842

Testing - Iteration 3

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
ProManageDAO	57.3 %	908	676	1,584
src	61.9 %	608	374	982
bu.met.cs.cs673.pm.dao	55.6 %	375	299	674
ProjectDAO.java	80.2 %	101	25	126
SessionFactorySingleton.java	91.2 %	31	3	34
StoryDAO.java	62.5 %	105	63	168
TaskDAO.java	44.4 %	56	70	126
UserDAO.java	37.3 %	82	138	220
bu.met.cs.cs673.pm.dto	75.6 %	233	75	308
Project.java	90.9 %	60	6	66
Role.java	0.0 %	0	3	3
Story.java	85.6 %	83	14	97
Task.java	68.2 %	45	21	66
User.java	59.2 %	45	31	76
junit	49.8 %	300	302	602

Project Management - Role Assignment

	ProManage User Interface	**ProManage DOA and REST Service**
User	Daniel C.	Olufemi
Project	Mariana A.	Luis
Story	Nethra	Daniel A.
Task	Roberto	Ray
Daniel A and Luis work on SQL tables		

Integration: We employ bottom-up integration testing in which each pair working on subsystem integrate first before integrating the whole project.

Project Management - Risk

Risk	Management of the Risk
Operational Risk: Most important is inadequate knowledge of tools needed for the project. This tools include Mybatis frameworks, Github, Restful Service etc	Other group member come to rescue as needed
Schedule Risk: Possibility that tasks assigned are not completed before the deadlines.	Group leader checking on members to make sure they are on track. We encourage each others to communicate any issues or request for help as early as possible
Requirement Inflation: As the project progresses more and more features that were not identified at the beginning of the project emerge that threaten estimates and timelines.	Priority to to complete the initial requirement. Then add additional requirements if we have time.
Poor Productivity and Group Member Turnover: Possibility that code written by a group member is not good enough or a member assigned a task drop out of the group	As soon as codes are check in, other members makes sure that the codes do what it is suppose to do. Also no member drop out, though we plan to distribute the load among remaining members.

Project Management - Risk

RISK	MANAGEMENT
Team members code fails to conform to established coding standards	Team review code to ensure conformity to coding standards
Improper choice and configuration of IDEs, Tools and frameworks	Team periodically review working tools to ensure appropriate choice of configuration tools and frameworks
Completed product may not meet acceptance test	Present artifacts produced during each iteration for user acceptance test and review
All requirements may not be implemented	Trace requirements implemented to pivotal tracker for completeness
User interface design elements implemented at the client tier may not be user friendly	Review and test sample user interfaces with users to ensure ease of use
Security credential management fails to secure user identity	Appropriate security tools will be implemented to secure user identity information
Unauthorized users may gain access to Promanage system	Identity management will be implemented to identify, authenticate and authorize users into the system

Project Management

Software Engineering Process

Planning Phase	Requirement analysis, identifying tool needed for the project
Iteration 1	Role assignment. Develop database schema, User Interface and Data Access Object for User, Project, Story,
Iteration 2	Modify database schema to include relationship and add Member, Role and, User_Role tables Develop service objects (Restful service). Integration of each subsystem(User UI - User DAO and Restful service, for example)
Iteration 3	Full integration of all the subsystems. Testing of ProManage to make sure it conforms to requirement analysis. Deployment on web service hosting (AWS for example).

Project Management - Metric

Hours (%)

Learning	Requirement Analysis	Design	Implementation	Testing	Communication	Unclassified
25	6	7	34	5	15	9

Hours by Iteration

Planning Phase	Iteration 1	Iteration 2	Iteration 3	Total
96	138.5	153	126.85	514.35
19%	27%	30%	25%	100%

Project Management - Metric

Hours by Members(%)

Member	Learning	Req. Analysis	Design	Implementation	Testing	Communication	Unclassified
1	12	14	11	30	7	16	10
2	23	8	10	28	10	14	8
3	25	6	0	37	0	20	12
4	25	3	4	43	6	11	7
5	26	6	0	35	0	19	14
6	44	3	27	14	3	10	0

Project Management - Metric

- LOC(UI, DAO and REST Service): 4306
- User Stories: 24 accepted
- Average hours per user story: 179
- Hours per week: 43
- Velocity: 10
- Complexity:
  - Nested block depth is 3(max. is 5)
  - McCabe Cyclomatic Complexity is 5(max. is 10)
- Readability and Maintainability:
  - We make sure that none of our methods has LOC greater than 50.
  - Also none of our classes has LOC greater than 750.

Project Management - Issues

Some of the issues we encountered

- Uncommitted and incorrectly loaded jars.
- Validation of “username” during user registration
  - Resolved by creating a servlet package.
- The database table names were not case sensitive in windows but turned out to be when we deployed to linux. Fixed it by correcting case.
- We had problem in login from UI to REST service
  - Resolved by merging UI into REST service
- We cannot delete a project because of the dependencies of stories and tasks
  - Resolved by just closing the project. After closing a project, it will exist in the database but it can not be modified.

Configuration Management - Tracking & Handling

PivotalTracker - Helps us to keep track of

- User requirements,
- Assignment of tasks and
- Completion of tasks.

GitHub - Helps us to handle

- Handle development process
- Code maintenance
- Version control

Configuration Management- Technologies

Challenges Faced

- Larger Group - Time constraint & coordination
- AWS is not intuitive
- Security was very difficult to implement
- Using GitHub
- Technologies were new to most of group members

Lessons Learnt

- Communication
- Team Work
- Co-ordination of work

SOFTWARE DEMO