SAN FRANCISCO STATE UNIVERSITY

FLORIDA ATLANTIC UNIVERSITY

SOFTWARE ENGINEERING

CSC 640/848, CEN 4010 – FALL 2008

INFINITY METRICS

AUTOMATIC COLLABORATION METRICS FOR JAVA.NET PROJECTS

TEST PLAN SPECIFICATIONS MILESTONE 4

Authors

Group 8

Marcello de Sales (msales@sfsu.edu)
Andres Ardila (aardila1@fau.edu)
Brett Fisher (bfisher@sfsu.edu)
Marilyne Mendolla (mmendoll@fau.edu)
Gurdeep Singh (gurdeep@sfsu.edu)

Advisors

Dr. Dragutin Petkovic (SFSU), Gary Thompson (SFSU),
Dr. Shihong Huang (FAU)

San Francisco, CA
Boca Raton, FL

December 08, 2009

Document ID

Date	12/10/2008
Responsible	Marcello de Sales
Document ID	INFINITYMETRICS_RS_20081210
Location	https://ppm-8.dev.java.net/milestones/4/INFINITYMETRICS_M4.doc

Revision History

Date	Version	Authors	Description
2008-09-28	0.1	Marcello de Sales	Template Creation
2008-12-08	0.2	Marcello de Sales	Test Plan Specification
2008-20-08	0.3	Marcello de Sales	Adding more test cases for the Usability Tests

Table of Contents

1. INTRODUCTION

This document describes the Quality Assurance Process for Infinity metrics. This chapter describes the purpose of this document, the scope of testing, audience and acronym.

1.1. PURPOSE

Infinity Metrics is an online web application that is loosely integrated with Java.net portal, when it comes tor reusing user's profile and data produced. In this way, the purpose of tests will be to exercise the two main integration components: The registration phase, which requires the use of Java.net username and passwords, and the login phase, which reuses the profile data acquired from Java.net, categorizing and directing the different type of users to its correct landing pages.

1.2. SCOPE

The tests approaches are in two flavors: Usability Tests, where regular users, described in the intended audience, uses the system without any support, as well as Functional Tests required for the same set of tools. The set of tests for the former is performed manually by the users, while the Functional Tests are automated tests written in PHP Unit Tests.

The tests **only include**

1. the registration and login of Students and Instructors, functionalities from the User Management component; which gets tightly integrated with the Metrics Workspace and Workspace Reports component.
2. In this way, it goes up to the landing page after the user performs a login.

Whereas this test plan includes most of the users management component, this test plan **does not include**

1. tests for the regular Java.net users, as well as the Metrics Reports integration, which would exercise details on the Metrics Workspace.

1.3. INTENDED AUDIENCE

The users of this document include Instructors and Students that have an account with Java.net portal. Regular users from Java.net **are not** included for this test plan.

1.4. DOCUMENT TERMINOLOGY AND ACRONYMS

User	Derived from the Actors, a user has the username and password from java.net. It will be used by the specialized users Instructor and Student for identity verification.
Student	Students play a secondary role on Infinity Metrics where he/she just fills out the participation form with his/her personal information collected from java.net. During registration, students can be identified as a team leader case he/she is a project owner of a java.net project. They are shown a Project Report as their participation.

Instructor	Instructors are identified having project ownership of any java.net project and is shown the collection of Metrics Workspace.
Personal Agent	The user's personal agent is the internet crawler that takes the user's username and password to navigate through java.net on his/her behalf to collect their information.
Metrics Workspace	The Metrics Workspace is the placeholder of a set of projects related to a given school term. In this way, the instructor has to create a workspace for each set of subprojects on java.net.
java.net Project	The java.net project is an existing project where users has membership with. This project is the unit that is tracked on the metrics. It is composed by a set of java.net usernames in the form of the student names.
Event	Each item on a java.net project RSS feed is called an Event, that is, by reading a RSS output. Each event is part an event category such as the commit message, an issue, and a topic posted in a discussion forum or an email sent to a given mailing list. Finally, it contains a reference to the creator of the item, who can be a student or a professor. It can also refer to a given entry by the instructor. This information is associated with the java.net username used in the RSS 'creator' tag.
Event Category	This is an enumeration of the types of events, or from which category the RSS item originated. It will be associated with each event and will also include the customized values of events.
Workspace Status	It is associated with the Metrics Workspace. Active is defined by default when the instructor creates the workspace environment. In that way, the instructor can tell his Personal Agent to collect metrics online.
Custom Event	Any situation that may arise among members of the same team throughout the semester.
Custom Event Tracker	The Custom Event Tracker enables an instructor to track Custom Events within a team. It allows Instructors to add entries about Custom Events, and catalogues them for monitoring.
Custom Event Entry	Custom Event Entries are entered by the Instructor to track significant events of a particular Custom Event in a team. An Instructor can add one or more entries for every instance of a Custom Event, which will be catalogued in the Custom Event Tracker.

2. Evaluation Mission and Test Motivation

The Usability and Functional Tests are necessary for validation of latest version of the requirements, as well as validate the current functionalities developed. In general, we want to know if Infinity Metrics meet the main user's expectance in the context of obtaining good feedback.

2.1. Background

Infinity Metrics has become a fairly complex system, since it's an integration with the existing java.net portal. In this way, problems related to the validation of user's identity and the collections of Events have been solved through the use of personal agent technology, which were exclusively developed to address data exchange from java.net in a transparent way. Please, refer to the latest version of the Milestone 2 documentation at http://ppm-8.dev.java.net/milestones/2/INFINIYNET_DOC, or browse through the Feature Issues at http://ppm-8.dev.java.net/issues.

2.2. Evaluation Mission

There are two main missions of this test plan: the Usability Tests are necessary to identify possible user interface mistakes related to design, positioning of graphs, and if the user has necessary information to operate the application with minimum training. While Usability Tests test users, the Functional Tests incorporate a level of detail that concentrates on the system's integration of different components.

2.3. Test Motivators

The tests presented are firstly motivated by the defined Use Cases, and therefore, will drive most of the user's actions. Other motivators include early risk mitigation, functional and non-functional requirement verification, systems integration, etc.

2.4. Target Test Items

The listing below identifies those test items for the software being written, as well as the supporting product elements that have been identified as targeting for testing. This list represents what items will be tested.

- Personal Computer or Macintosh computer with Internet connectivity;
- Mozilla Firefox Internet Browser, Opera, Google Chrome, Microsoft Internet Explorer or other.

3. USABILITY TEST PLAN

This section describes the usability test plan, which includes sections covering the test objectives, the test plan and a questionnaire form for graded responses.

3.1. TEST OBJECTIVES

This test evaluates how the user perceives and executes our software without any additional training.

3.2. USABILITY TEST PLAN

3.2.1. TC001 – CREATE AN ACCOUNT AS AN INSTRUCTOR

3.2.1.1. TC001 – PURPOSE

This test case evaluates how the users can register as an Instructor of a given institution. The online format is located at https://ppm-8.dev.java.net/issues/show_bug.cgi?id=181

3.2.1.2. TC001 – PRECONDITIONS SETUP

1. Have an account on java.net;
2. Have the role of "Project Owner" of a java.net Project containing subprojects, which represents a given course structure for a given term; For example

http://ppm.dev.java.net is a parent project containing subprojects ppm-1 through ppm-13.

3.2.1.3. TC001 – TEST PROCEDURE

Steps to Reproduce	Expected Results
1.S. Go to http://www.infinitymetrics.net/rc1	1.R. The Infinity Metrics page is shown.
2.S. Try Registering in the system as an Instructor. If your institution is not listed, you can optionally your institution.	2.R. Infinity Metrics system identifies you as an instructor of a given project; 3.R. Shows you a registration confirmation message after the confirmation; 4.R. You receive an email confirmation with follow-up information.

3.2.1.4. TC001 – EVALUATION QUESTIONNAIRE

1. I could successfully register as an instructor for the institution I'm related to.
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

2. The system could successfully handle possible error messages and information messages
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3. There are enough information to perform this action without any additional training or manual
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

4. I had enough textual information about what I had to do to complete the task
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

5. The follow-up messages are complete and accurate (any email received from Infinity Metrics)
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3.2.2. TC002 – Create an account as a Student

3.2.2.1. TC002 – Purpose

This test case evaluates how the users can register as a student of a given institution.

3.2.2.2. TC002 – Preconditions Setup

1. Have an account on java.net;
2. Have the role of "Developer" or/and "Project Owner" of a java.net Project that is contained as subproject of a main project.

http://ppm-8.dev.java.net is a subproject contained by a parent project called ppm.

3.2.2.3. TC002 – Test Procedure

Steps to Reproduce	Expected Results
1.S. Go to http://www.infinitymetrics.net/rc1	1.R. The Infinity Metrics page is shown.
2.S. Try Registering in the system as an Student. You must be able to identify your institution, since it's assumed that your instructor has already setup the environment for your institution.	2.R. Infinity Metrics system identifies you as a "Team Lead" in case you have a "Project Owner" role in a project, or a "Team Member" if not; 3.R. Shows you a registration confirmation message after the confirmation; 4.R. You received an email confirmation with follow-up information.

3.2.2.4. TC002 – Evaluation Questionnaire

1. I could successfully register as an instructor for the institution I'm related to.
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

2. The system could successfully handle possible error messages and information messages
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3. There are enough information to perform this action without any additional training or manual
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

4. I had enough textual information about what I had to do to complete the task
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

5. The follow-up messages are complete and accurate (any email received from Infinity Metrics)
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3.2.3. TC003 – Login into The System as an Instructor

3.2.3.1. TC003 – Purpose

This test case evaluates how the login into the system as an Instructor. The online version is located at https://ppm-8.dev.java.net/issues/show_bug.cgi?id=181

3.2.3.2. TC003 – Preconditions Setup

1. Successfully execute the test case TC001.

3.2.3.3. TC003 – Test Procedure

Steps to Reproduce	Expected Results
1.S. Go to http://www.infinitymetrics.net/rc1	1.R. The Infinity Metrics page is shown.
2.S. Try logging-in into the system with your current java.net username and password used during the registration.	2.R. The Infinity Metrics system identifies your type of user Instructor; 3.R. You can see your Metrics Workspaces Collection page; 4.R. The Infinity Metrics system can offer you to create a new Workspace based on the project used on the registration.

3.2.3.4. TC003 – Evaluation Questionnaire

1. I could successfully register as an instructor for the institution I'm related to.
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

2. The system could successfully handle possible error messages and information messages
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3. There are enough information to perform this action without any additional training or manual
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

4. I had enough textual information about what I had to do to complete the task
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

5. The follow-up messages are complete and accurate (any email received from Infinity Metrics)
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3.2.4. TC004 – Login into The System as a Student

3.2.4.1. TC004 – Purpose

This test case evaluates how the login into the system as a student.

3.2.4.2. TC004 – Preconditions Setup

1. Successfully execute the test case TC002.

3.2.4.3. TC004 – Test Procedure

Steps to Reproduce	Expected Results
1.S. Go to http://www.infinitymetrics.net/rc1	1.R. The Infinity Metrics page is shown.
2.S. Try logging-in into the system with your current java.net username and password used during the registration.	2.R. The Infinity Metrics system identifies your type of user as Student; 3.R. You can see the Project Report; 4.R. The Infinity Metrics system shows that you are a Team Leader, in case you have the role "Project Owner".

3.2.4.4. TC004 – Evaluation Questionnaire

1. I could successfully register as an instructor for the institution I'm related to.
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

2. The system could successfully handle possible error messages and information messages
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3. There are enough information to perform this action without any additional training or manual
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

4. I had enough textual information about what I had to do to complete the task
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

5. The follow-up messages are complete and accurate (any email received from Infinity Metrics)
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3.2.5. TC100 – Instructor Creates a Workspace

3.2.5.1. TC100 – Purpose

This test case evaluates how an Instructor creates a new metrics workspace (a configuration for a set of projects). The online version of this test case is located at
https://ppm-8.dev.java.net/issues/show_bug.cgi?id=182

3.2.5.2. TC100 – Preconditions Setup

1. Login into the system as an instructor (Execute test case TC003 https://ppm-8.dev.java.net/issues/show_bug.cgi?id=181)

3.2.5.3. TC100 – Test Procedure

Steps to Reproduce	Expected Results
1.S. Choose the option to create a new Metrics Workspace	1.R. After giving the needed information, your Metrics Workspace should show the list of projects related to the parent project.
2.S. Click on the Metrics Reports Icon.	2.R. Verify that the newly created workspace is shown in the list of metrics workspace on the initial page.

3.2.5.4. TC100 – Evaluation Questionnaire

1. I could successfully create a new workspace (configuration) for a given parent project.
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

2. The system could successfully handle possible error messages and information messages
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3. There are enough information to perform this action without any additional training or manual
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

4. I had enough textual information about what I had to do to complete the task
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

5. The follow-up messages are complete and accurate (any email received from Infinity Metrics)
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3.2.6. TC300 – Instructor View Workspace Report

3.2.6.1. TC300 – Purpose

This test case evaluates how an Instructor views the reports for a given workspace (a configuration for a set of projects). The online version of this test case is located at https://ppm-8.dev.java.net/issues/show_bug.cgi?id=183

3.2.6.2. TC300 – Preconditions Setup

1. Create a new Workspace configuration (successfully execute the test case TC100).

3.2.6.3. TC300 – Test Procedure

Steps to Reproduce	Expected Results
1.S. Click on the Reports icon	1.R. Verify that the list of metrics workspace is shown.
2.S. Click on the name of a given workspace you have in your list.	2.R. Verify that the report of the report is shown in the front page; 3.R. Verify that the list of projects related to the given report is visible; 4.R. Verify that the title and description of the workspace is visible.

3.2.6.4. TC300 – Evaluation Questionnaire

1. I could successfully view the report for a given workspace I have previously created.
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

2. The system could successfully handle possible error messages and information messages
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3. There are enough information to perform this action without any additional training or manual
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

4. I had enough textual information about what I had to do to complete the task
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

5. The follow-up messages are complete and accurate (any email received from Infinity Metrics)
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3.2.7. TC301 – Instructor View Project Report

3.2.7.1. TC301 – Purpose

This test case evaluates how an Instructor view the reports for a given project.

3.2.7.2. TC301 – Preconditions Setup

1. View the reports of an existing workspace; execute test case TC300 https://ppm-8.dev.java.net/issues/show_bug.cgi?id=183

3.2.7.3. TC301 – Test Procedure

Steps to Reproduce	Expected Results
1.S. From the reports of a given workspace, click in a given project name from the list of projects.	1.R. Verify that the report is shown for the given project, which contains the name of the project; 2.R. Verify that the list of members of the project is visible 3.R. Verify that the name and the description of the project is visible;

3.2.7.4. TC301 – Evaluation Questionnaire

1. I could successfully view the report of a given project chosen from a workspace.
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

2. The system could successfully handle possible error messages and information messages
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3. There are enough information to perform this action without any additional training or manual
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

4. I had enough textual information about what I had to do to complete the task
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

5. The follow-up messages are complete and accurate (any email received from Infinity Metrics)
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3.2.8. TC302 – Instructor View Student Report

3.2.8.1. TC302 – Purpose

This test case evaluates how an Instructor view the reports for a given student.

3.2.8.2. TC302 – Preconditions Setup

1. View the reports of an existing project; execute test case TC301
https://ppm-8.dev.java.net/issues/show_bug.cgi?id=184

3.2.8.3. TC302 – Test Procedure

Steps to Reproduce	Expected Results
1.S. From the report of a given project, click in a given member name from the report of that given project	1.R. Verify that the report is shown for the given user; 2.R. Verify that there is a link to the user profile.

3.2.8.4. TC302 – Evaluation Questionnaire

1. I could successfully view the report for a given student chosen from the project workspace
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

2. The system could successfully handle possible error messages and information messages
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

3. There are enough information to perform this action without any additional training or manual
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

4. I had enough textual information about what I had to do to complete the task
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

5. The follow-up messages are complete and accurate (any email received from Infinity Metrics)
   [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Disagree

4. FUNCTIONAL TEST PLAN

This chapter describes the purpose of the Functional Tests performed in the system. As mentioned in the first chapters, the functional tests are automatically executed by the integrated development environment. Although the entire system has Unit Tests and System Tests, we will present here the Functional Tests developed using PHPUnit.

4.1. TEST OBJECTIVES AND APPROACH

These tests exercise the functional and non-functional requirements modeled in the Use Cases, and therefore, they are designed as Black-Box tests. The evaluation process will be testing if the integration of each of the components defined in each Component's Controller is working as described by the Use Cases.

The approach is to map each test into Test Suites related to each User Case, resenting them as a PHPUnit class, based on specialization. Taking advantage of its capabilities similar to JUnit, the setup and tear down methods are used to correctly have the environment on the correct state, while the create test methods that exercise different sections of the component are written with the prefix **test**. Each team member of the PPM-8 project developed, among other types of tests, Functional tests for each class of the system. The complete list of tests for the UserManagement Controller, for example, is located at

https://ppm-8.dev.java.net/source/browse/ppm-8/trunk/app/classes/infinitymetrics/tests/functional/user/

Here's the list of test methods that exercise the UserManagementComponent.class.php, which implements the Use Cases described. The third column of each table describes the Test Class file under the functional test subversion folder given above.

UC001 Functional Tests	**Student Registration**	**UC001Test.class.php**
[x] Successful student registration	public function testSuccessfulStudentRegistration() {..}	
[x] Missing fields for student registration	public function testMissingFieldsStudentRegistration() {..}	
[x] Register existing student leader registration	public function testRegisterExistingStudentLeaderRegistration() {..}	

UC002 Functional Tests	**Instructor Registration**	**UC002Test.class.php**
[x] Successful instructor registration	public function testSuccessfulInstructorRegistration() {..}	
[x] Missing fields for instructor registration	public function testMissingFieldsInstructorRegistration() {..}	
[x] Register existing instructor registration	public function testRegisterExistingInstructorRegistration() {..}	

UC003 Functional Tests	**Users Login**	**UC003Test.class.php**
[x] Valid student login	public function testValidStudentLogin() {..}	
[x] Valid instructor login	public function testValidInstructorLogin() {..}	

[x] Wrong fields for student login	public function testWrongFieldsStudentLogin() {..}
[x] Wrong fields for instructor login	public function testWrongFieldsInstructorLogin() {..}
[x] Missing fields for student login	public function testMissingFieldsStudentorLogin() {..}
[x] Missing fields for instructor login	public function testMissingFieldsInstructorLogin() {..}

4.2. Functional Test Plan

This test plan will only include the plan for the student, since the ones for the Instructor and login follow the same pattern.

4.2.1. UC001Test – Create an account as a Student

4.2.1.1. UC001Test – Purpose

This test case evaluates how the users can register as an Instructor of a given institution. It verifies different scenarios such as the successful instructor registration and the exceptional attempts of creating a new instructor with missing values or trying to register the same user more than once.

4.2.1.2. TC001 – Preconditions Setup

1. Clean up the database system;
2. Create a new institution identified by a given abbreviation;
3. Create a new project identified by a java.net project name;

http://ppm.dev.java.net is a parent project containing subprojects ppm-1 through ppm-13.

4.2.1.3. TC001 – Test Procedure

Successful student registration

Steps to Reproduce	Expected Results
1.S. Register a new Student with valid data, associating him/her with the Institution and the Project instances created on the Setup phase.	1.R. A new instance of Student is created.
2.S. Assert that the object created is not null and is of type Student.	2.R. Assertions correctly return true
3.S. Verify that you can retrieve the newly created Student by using its relation with the Institution.	3.R. The Student object is retrieved from the relation with the Institution.
4.S. Verify that the users properties are not null, including the institution identification.	4.R. All the properties are correctly loaded, including the institution identification.
5.S. Verify that the you can retrieve the newly created Student by using its relation with a the	5.R. The Student object is retrieved from the relation with the project.

java.net project name.	

Missing Fields for student registration

Steps to Reproduce	Expected Results
1.S. Try registering a new Student with invalid data, associating him/her with the Institution and the Project instances created on the Setup phase.	1.R. An exception is thrown and the new Student was not created on the database; 2.R. The error message indicates that the input is incorrect; 3.R. Also very that the database doesn't contain the object that was attempted to be saved.

Register Existing Student leader student registration

Steps to Reproduce	Expected Results
1.S. Try registering an existing Student with valid data, associating him/her with the Institution and the Project instances created on the Setup phase.	1.R. An exception is thrown and the new Student was not created on the database; 2.R. The error message indicates that the Student already exists with the given information. 3.R. Also very that the database doesn't contain the object that was attempted to be saved.

4.2.1.4. TC001 – Tear Down

1. Clean up the database system;

4.2.1.5. TC001 – Test Class Representation

A complete version of the test script is located at

https://ppm8.dev.java.net/source/browse/ppm8/trunk/app/classes/infinitymetrics/tests/functional/user/UC001Test.class.php

```
class UC001Test extends PHPUnit_Framework_TestCase {

    private $institution;
    private $project;
    private $userTypeEnum;

    public function  __construct() {
        $this->userTypeEnum = UserTypeEnum::getInstance();
    }

    private function cleanUpAll() {
        PersistentUserXProjectPeer::doDeleteAll();
        PersistentUserPeer::doDeleteAll();
        PersistentInstitutionPeer::doDeleteAll();
        PersistentProjectPeer::doDeleteAll();
    }

    protected function setUp() {
        parent::setUp();
        $this->cleanUpAll();

        $this->institution = new Institution();
        $this->institution->setName('San Francisco State University');
        $this->institution->setAbbreviation("SFSU");
        $this->institution->setCity('San Francisco');
        $this->institution->setStateProvince('CA');
        $this->institution->setCountry('USA');
        $this->institution->save();
```

```
    $this->project = new PersistentProject();
    $this->project->setProjectJnName("ppm-8");
    $this->project->setSummary("Infinity Metrics");
    $this->project->save();
}

public function testSuccessfulStudentRegistration() {
    try {
        //Saving the student leader
        $createdStudent = UserManagementController::registerStudent(
                            "username2", "password", "email@gmail.com", "firstNameLeader",
                            "lastNameLeader", "909663916", $this->project->getProjectJnName(),
                            $this->institution->getAbbreviation(), true);
        $this->assertNotNull($createdStudent, "The registered student is null");
        $this->assertEquals($this->userTypeEnum->STUDENT, $createdStudent->getType(), "The registered user is not
                                                                                an instance of Student");

        $studentInstitution = PersistentUserXInstitutionPeer::retrieveByPk($createdStudent->getUserId(),
                                                                $this->institution->getInstitutionId());
        $this->assertNotNull($studentInstitution, "The user x institution relation was not created for the student.");
        $this->assertEquals("909663916", $studentInstitution->getIdentification(), "The student school
                                                                    identification is incorrect");
        $this->assertEquals($createdStudent->getUserId(), $studentInstitution->getUserId(), "The user id is
                                                incorrect on user x institution for the student");
        $this->assertEquals($this->institution->getInstitutionId(), $studentInstitution->getInstitutionId(),
                                        "The institution id is incorrect on user x institution for the student");

        $stXProjec = PersistentUserXProjectPeer::retrieveByPK($createdStudent->getJnUsername(),
                                                                $this->project->getProjectJnName());
        $this->assertNotNull($stXProjec, "The relationship between student and project was not created");
        $this->assertEquals($this->project->getProjectJnName(), $stXProjec->getProjectJnName(), "The project name
                                                                    is incorrect on user x project");
        $this->assertEquals($createdStudent->getJnUsername(), $stXProjec->getJnUsername(), "The java.net
                                                username is incorrect on user x project for student");
        $this->assertTrue($stXProjec->getIsOwner() == 1, "The student is a leader, and therefore, a project owner.");

    } catch (InfinityMetricsException $ime){
        $this->fail("The successful login scenario failed: " . $ime);
    }
}
```

4.2.1.6. TC001 – Test Results and Log

As part of the PHPUnit framework, the results and test log is generated in XML. Here's the result of running the entire file Clean up the database system;

```
<testsuite name="UC001Test"
           file="C:\ppm8-dev\app\classes\infinitymetrics\tests\functional\user\UC001Test.class.php"
          tests="3" assertions="30" failures="0" errors="0" time="1.514736">
  <testcase name="testSuccessfulStudentRegistration"
           class="UC001Test"
           file="C:\ppm8-dev\app\classes\infinitymetrics\tests\functional\user\UC001Test.class.php"
           line="78" assertions="20" time="0.676405"/>
  <testcase name="testMissingFieldsStudentRegistration"
           class="UC001Test"
           file="C:\ppm8-dev\app\classes\infinitymetrics\tests\functional\user\UC001Test.class.php"
           line="150" assertions="9" time="0.223845"/>
  <testcase name="testRegisterExistingStudentLeaderRegistration"
           class="UC001Test"
           file="C:\ppm8-dev\app\classes\infinitymetrics\tests\functional\user\UC001Test.class.php"
           line="202" assertions="1" time="0.614486"/>
</testsuite>
```

4.2.1.7. TC001 – Test Coverage

Our tests also include Test Coverage to find out how good the tests were developed. For the User Management controller, we could achieve 72.66% of Line Coverage (287/395 lines) for the entire class, and 100% of Functional Coverage (3/3) methods related to the Student Registration. Please refer to the appendices for more information.

5. Peer Code Review

This section describes the strategy for peer code review.

5.1. Purpose

Given that open-source software projects have the characteristic of shared ownership, the process of peer code review is done in a very easy way. The purpose of Peer Review is to avoid team members committing code to the repository without being effectively reviewed and tested by other peers.

5.2. Peer Review Process

After the team members have been assigned to develop tasks with specified Issue Tracking system, the team members will be creating TASKS. For a given task, a functionality is “attacked” and the development is tracked in that single place.

After the development of a particular functionality, the team member is responsible for creating a PATCH that represents the difference between the current version on the Subversion repository and its functionality. In this way, members can assign the Issue to a given a given peer to apply the patch and verify the functionality described in the Issue Tracker. After the revision has been done, the reviewer can re-assign the issue back to the owner of the Issue with the results.

6. RISK ASSESSMENT

The risk management is an important tool to identify possible problems and how to mitigate them. It is used by the entire team, especially by the project leaders. The Risk Analysis subsection describes each of the identified risks for this project, defining its scope and type. Along with that, the subsection Risk Monitoring describes probabilities of such risks to take place, indicators of that show when the risk has emerged the strategies on how to mitigate each of them.

6.1. RISK ANALYSIS

Risk	Risk Type	Risk Scope	Description
Team Member Skill Set	People	Project/Product	Team members do not possess the programming skill set to adequately complete the project.
Scheduling Difficulties	Organizational	Project	Time zone differences, schedule overload and conflicting schedules among team members will occur
Legal Liabilities	Business Environment	Business	The Terms of Use in java.net do not permit automated or scripted access to the site.
Underestimating Project Size	Estimation	Project/Product	The team will underestimate the size of the project during the planning stage
Requirement Changes	Requirements Risk	Project/Product	The requirements will change after the requirement planning phase
Development IDE Underperformance	Tools	Project	The tools used to develop the software will not perform as expected (i.e. NetBeans 6.5 is a Beta release)
java.net updates its software platform	Technology	Project/Product	java.net updates the version of CEE to a new version that changes the layout of the HTML at the sections where the personal agent uses.
Team Member Drop-out	People	Project	Team members will withdraw from the class and thus out of the project
Team Semester Completion Conflict	People	Project/Product	FAU semester ends 2 weeks earlier than SFSU; FAU team members will end participation in project after Milestone 3.
Team Member Lack of Participation	People	Project	Team members will not participate sufficiently to fulfill the project's requirements.
Deployment Platform Malfunction	Technology	Project	The deployment environment (T2000) server will malfunction and render our system inoperable or inaccessible.
Support software underperformance	Tools	Project	The distributed virtual development environment will not perform as expected (i.e. Virtual Machine)
Underestimating Development Time of Requirements	Estimation	Project/Product	The time needed to develop and implement the requirement specifications will exceed the length of the semester.
Group Dynamics Conflicts or Failure	People	Project	The team will reach an impasse or a irreconcilable differences which will hinder the progress of the project.
CASE Tool Underperformance	Tools	Project	The code generated by CASE tools is insufficient or inaccurate

6.2. Risk Monitoring

Risk	Probability	Effects	Potential Indicators	Mitigation Strategy
Team Member Skill Set	High	Serious	- Constant build breaks, inability to meet; - Implementation deadlines	- Have team members follow tutorials and training sessions to enhance skill set.
Scheduling Difficulties	High	Serious	- Missed meetings - Poor communication, - poor performance and missed deadlines in project & other activities	- Increase communication; - Allow for flexibility in case of conflict; - Provide time management tools for team members to fulfill all demands on time effectively and efficiently.
Legal Liabilities	Low	Armageddon	Legal suits are filed, cease & desist orders, etc.	- Research legal Terms of Use regarding use & access policies on java.net.
Underestimating Project Size	Moderate	Serious	Issues queue accumulate, and team seems unable to grasp scope of project design.	- Be realistic about time allowance for development; - Use good planning strategies.
Requirement Changes	Moderate	Serious	- Multiple revisions to requirements due to customer requests; - re-writing previously developed components to accommodate new requirements.	- Perform requirement validation; - Verification techniques along with traceability.
Development IDE Underperformance	Moderate	Serious	Constant application crashes, loss of unsaved code.	- Have version control in case of file corruption; - Stay up to date on NetBeans released/bug fixes and periodically download last build of IDE.
Technology Change	Low	Armageddon	Personal Agent unable to retrieve RSS feeds, project sets, or metrics	Stay up to date with java.net forums and news.
Team Member Drop-out	Moderate	Tolerable	Team members no longer in class	Build strong team cohesiveness.
Team Semester Completion Conflict	High	Tolerable	FAU Team will complete their semester two weeks prior to SFSU completion	- FAU team will complete their components to the best of their abilities - Extensive documentation and communication of FAU work contributed so far on project - FAU is to submit a document to SFSU with details about what components are complete, which are not, and what remains to be implemented.
Team Member Lack of Participation	Moderate	Tolerable	Lack of participation, missed meetings, lack of communication	Encourage team members to contribute to the best of their ability and allow flexibility in contribution levels.
Deployment Platform Malfunction	Low	Armageddon	System is inoperable or offline.	Have a set of policies & procedures with contact information for technical support team responsible for maintaining the T2000 server.
Support software underperformance	Moderate	Serious	Inability to use virtual environment.	Test environment thoroughly and stabilize environment before distributing to team.
Underestimating	Moderate	Serious	End of semester with no	Carefully plan execution time for

Development Time of Requirements			complete implementation	each requirement. Prioritize requirements. Keep time constrains in mind.
Group Dynamics Conflicts or Failure	Low	Serious	Team Custom Events and communication ceases.	Provide a positive communication environment, morale-boosting techniques and conflict resolution.
CASE Tool Underperformance	Low	Insignificant	Code breaks and behaves unexpectedly.	Test thoroughly code generated by CASE tools and allow sufficient time for re-writes.

Open Risks:

Risk	Potential Indicators	Effects	Team Experience	Mitigation Strategy
Team Member Skill Set	- Constant build breaks, - Inability to meet Implementation deadlines	Serious	- Team members have been submitting working code for their components	- No longer an open risk.
Scheduling Difficulties	- Missed meetings - Poor communication, - poor performance and missed deadlines in project & other activities	Serious	- Acceptable communication between team members - Notification prior to missed meetings - Weekly meeting scheduled to work with all team member's schedule	- No longer an open risk
Legal Liabilities	Legal suits are filed, cease & desist orders, etc.	Armageddon	- Legal matters researched	- Still open risk pending deployment of InfinityMetrics
Underestimating Project Size	Issues queue accumulate, and team seems unable to grasp scope of project design.	Serious	-This risk has been largely experienced within the team - Requests have been made to simplify project - Decision has been made to move forward and implement as intended	- This is still an active issue within the team - Adequate communication of implementation problems
Requirement Changes	- Multiple revisions to requirements due to customer requests; - re-writing previously developed components to accommodate new requirements.	Serious	- The team has done some re-writes of requirements and use cases while implementing. - Customer requirements have remained stable	- No longer an open risk.
Development IDE Underperformance	Constant application crashes, loss of unsaved code.	Moderate	- Several computer crashes experienced by one team member. - Team does software updates and frequent SVN to alleviate this risk	- Still open risk - Effect changed to moderate due to SVN tracking
Technology Change	Personal Agent unable to retrieve RSS feeds, project sets, or metrics	Armageddon	- Personal Agent has been successfully implemented	- No longer an open risk.
Team Member Drop-out	Team members no longer in class	Tolerable	- We have lost one team member from the FAU team.	- Still an open risk
Team Semester Completion Conflict	FAU Team will complete their semester two weeks prior to SFSU completion	Tolerable	- Team has implemented components ahead of schedule to counteract this risk.	- Still open risk. - FAU team will complete their components to the best of their abilities - Extensive documentation and communication of FAU work contributed so far on project

				- FAU is to submit a document to SFSU with details about what components are complete, which are not, and what remains to be implemented.
Team Member Lack of Participation	Lack of participation, missed meetings, lack of communication	Tolerable	- This has been experienced within the team. - Encouraged of team members to participate. - Assign team members components to implement	- Still an open risk
Deployment Platform Malfunction	System is inoperable or offline.	Armageddon	- System has been launched on the T2000 server successfully.	- No longer an open risk.
Support software underperformance	Inability to use virtual environment.	Serious	- All team members have virtual box installed and operational.	- No longer an open risk.
Underestimating Development Time of Requirements	End of semester with no complete implementation	Serious	- Team leader has implementation schedule to keep team on track.	- Still an open risk.
Group Dynamics Conflicts or Failure	Team Custom Events and communication ceases.	Serious	- Team has developed a very professional working environment throughout this semester.	- No longer an open risk.
CASE Tool Underperformance	Code breaks and behaves unexpectedly.	Insignificant	- Team members have test cases for all their code. - Minor issues have been experienced, but they have all been resolved.	- No longer an open risk

7. APPENDIX

7.1. TEST COVERAGE RESULTS FOR USER MANAGEMENT CONTROLLER

The complete annotated results of the test coverage can be reviewed at

https://ppm-8.dev.java.net/milestones/4/ppm8_dev_app_classes_infinitymetrics_controller_UserManagementController.class.php.html