Test Plan

Co-op Evaluation System

Senior Project 2014-2015

Team Members:

Tyler Geery
Maddison Hickson
Casey Klimkowsky
Emma Nelson

Faculty Coach:

Samuel Malachowsky

Project Sponsors:

Jim Bondi (OCSCE)
Kim Sowers (ITS)

Table of Contents

Revision History

Version	**Primary Author(s)**	**Description of Version**	**Date Completed**
v1.0	Emma Nelson, Maddison Hickson, Casey Klimkowsky, Tyler Geery	Initial revision	January 30, 2015

1 Introduction

This is the Master Test Plan for the Co-op Evaluation System project. This plan will address only those items and elements that are related to the new Co-op Evaluation System. The primary focus of this plan is to ensure that the new Co-op Evaluation System provides the same level of information and detail as the current system while allowing for improvements in usability and other annoyances that currently exist with the system.

As required of us by ITS, all testing will take place on the DEV and TEST environments of the project. The DEV environment is used for unit testing, while the TEST environment is reserved for user acceptance testing and more rigorous system testing. The estimated development timeline for this project is very aggressive (approximately 3 months); as such, any delays in the development process could have significant effects on the test plan.

2 Features to be Tested

The following is a list of the areas to be focused on during testing of the new Co-op Evaluation System. The level of risk involved will be reevaluated as the development team moves farther along into the development process.

Feature	**Description**	**Level of Risk**
View/Edit/Save/Submit Evaluations	The fundamental feature of the system is editing and submitting co-op evaluations. student and employer users must be able to edit, save, and submit evaluations.	High
Searching of Evaluations	The admin and evaluator users must be able to search through the student/employer evaluations within the system using specific criteria. If an evaluation exists with that criteria, then it must be returned by the system.	High
Importing of Evaluations	The admin user must be able to import a file of evaluations into the system. Once these evaluations have been added, the user must be able to search for them within the system	High
Sending/Editing Emails	The admin user utilizes this functionality to send reminders to students and employers about completing their evaluations and other information required by the system.	Medium
Verifying Status of Emails Sent	An admin may want to ensure the email addresses in the system are correct. To verify this, they can check the status of sent emails using a list of all emails that failed to send.	Medium

Adding/Deleting Users	An admin requires the ability to add new users to the system and remove current users from the system.	Medium
Transferring User Privileges	An admin may transfer the privileges of one user to another user within the system.	Low
Adding Colleges into the System	An admin must have the capability to add a new college into the system in the event that a new college is introduced to RIT.	Medium
Adding Departments into the System	An admin must have the capability to add a new department into the system, in the event that a new department is introduced to RIT.	Medium
View/Edit/Save/Create Forms	An admin must have the ability to create new forms in the system, which are used as student work reports or employer evaluations. The admin can edit forms that already exist in the system. An admin can save a new or edited form.	High
Approve/Reject Student Work Reports	An evaluator must have the ability to either accept or reject a student’s work report once both the student and employer forms have been completed. Upon acceptance or rejection, an email will be sent to the student to inform them of their status.	Medium
Employer Login	Employers must be able to login using their email address and provided password since they cannot use Shibboleth.	Medium

3 Features Not to be Tested

The following is a list of the areas that will not be specifically addressed during testing of the application. All testing in these areas will be indirect as a result of other testing efforts in our project or in previous projects.

Shibboleth

RIT uses Shibboleth single sign-on in order to authenticate its users. Shibboleth provides our application with user information, such as username, full name, and what type of account they have (e.g. “faculty”, “staff”, or “student”). Although we will verify that our application responds appropriately to the information given to us by Shibboleth, we will not be testing Shibboleth itself. We will be under the assumption that the information given to us by Shibboleth is correct.

Database Code

Databases by nature are stateful, as the whole idea of a database is to manage state. It is difficult to manage state transactions using unit tests, and most meaningful state transitions span multiple database interactions [1]. Using this approach, we be able to achieve at least 70% code coverage, and our coverage will be on code that truly matters. Our database code is accessed by our services, and therefore will be tested indirectly by testing our services.

Accessors and Mutator Methods

Informally “getters” and “setters”, the sole purpose of these methods is to retrieve the value of a variable from an object. The functionality of our getters and setters will be tested indirectly by testing other, more complicated class methods. In general, there is no value in testing these methods directly; if there is added complicated functionality in a particular getter or setter, we will then write a respective unit test.

4 Approach (Strategy)

4.1 Testing Levels

Usability Testing

Usability testing will be completed by the development team. We will be testing at least one user from each of the four main user roles: student, employer, evaluator, and administrator. This way, we will be able to test all of the user interface screens of the application, and we can gather feedback from each type of user. This form of testing will first be completed on clickable mockups and later on functional code.

Unit Testing

Unit testing will also be completed by the development team. We will be aiming for 70% of code coverage on features listed above as “To Be Tested”. As each milestone is delivered, unit tests for that feature set will be delivered as well.

Acceptance Testing

Acceptance testing will be completed by the development team and project sponsors, particularly individuals from OCSCE. This form of testing will be carried out to ensure that the development team has, at a minimum, matched the functionality of the current system.

Load and Stress Testing

Any form of load or stress testing, or similar testing, will be performed by ITS. It is important to know whether the application can handle the expected number of concurrent users, as well as the upper limits of capacity within the system; however, we believe it would be beneficial for ITS to perform this testing instead of the development team, as it is likely they already have a process for completing these forms of testing in place.

4.2 Configuration Management/Change Control

GitHub

The application is hosted on GitHub; therefore, git will be our version control system. Additionally, we will use the Issues feature of GitHub for keeping track of existing bugs and features yet to be implemented. All of our unit tests will be published to GitHub as well.

Trello

Trello is used to track our high-level development milestones, and by association, our major unit testing milestones. We use Trello to manage any related feature-level or process-oriented tasks that may arise as well.

Slack

Slack is our primary communication tool. We use Slack to update each other on our progress with the project; therefore, we will use it to update each other on our progress with testing as well. Slack integrates with Trello, so we get live updates when another team member performs an action on Trello, such as creating a new card or moving an existing card.

4.3 Test Tools

JUnit

JUnit (version 4.11) will be used to test our code written in Java. JUnit will be primarily used to test our controllers and services (e.g. UserAuthenticationService).

Jasmine

Jasmine (version 2.1.3) will be used to test our code written in JavaScript. In particular, Jasmine will be used mostly to test our code related to AngularJS (e.g. front-end controllers and services).

IntelliJ Coverage

We will be using the coverage tool built into IntelliJ in order to determine the code coverage achieved by our unit tests. Using this tool, coverage statistics will be available to us for each class and package, with an exact percentage of methods and lines of code covered by unit tests.

Google Drive

We will be using multiple tools from Google Drive for several different types of testing. For usability testing, we will be using a Google Form to collect feedback from users and to determine the SUS score. We will likely have a separate spreadsheet for tracking our observations as well.

We will also be using a spreadsheet within Google Drive for acceptance testing. Within this spreadsheet, there will be a row for each requirement in our Requirements Document, and a column in each of these rows indicating whether the requirement has been accepted or not.

4.4 Measures and Metrics

Code Coverage

We will measure code coverage to determine the degree to which the source code is tested by our unit tests. Code coverage is measured as a percentage of lines of code out of all lines of code that are covered by our test suite. As mentioned in Section 5.1, we are aiming for 70% of code coverage.

Acceptance Test Pass Rate

Tracking the number of acceptance tests passed over the number performed for each phase of testing (correlating to each role's development) will allow the development team to check for regressions in previously completed portions of the code at an interaction level. This number will be recorded each time the acceptance test suite is run in order to track the change in the number over the course of development. The ultimate goal is to reach a 100% pass rate.

SUS Score

The Software Usability Scale (SUS) provides a Likert-type measure of user satisfaction with a particular software product. A SUS score for our application will be generated using user feedback gathered during usability testing. The score may also be generated by role in order to determine which part of the product is struggling the most. Based on user responses, we may modify our wireframes prior to the start of implementation to fix any usability flaws that have been detected.

5 Item Pass/Fail Criteria

The pass or fail criteria for a test item is dependent on the type of testing being performed on that item.

Usability Testing

Usability testing is much more subjective than the other types of testing listed below. It is less about passing or failing as it is gleaning useful information to inform future design changes or lack thereof. It will be run in two of phases: one with clickable mockups and the other with working software. Usability testing will be considered minimally complete after testing one person in each of the user classes during each of the testing phases. Ideally we will test three or four users in each role in order to get a usable SUS score.

Unit Testing

The pass/fail criteria for unit tests is very straightforward, and will be determined by JUnit, our unit testing framework. After the execution of our unit test suite, JUnit will mark each individual unit test indicating whether it passed or failed according to our assert statements.

The entirety of our unit testing will be considered complete once all test cases pass successfully, and we have achieved at least 70% code coverage.

Acceptance Testing
For acceptance testing, each of our requirements from our Requirements Document will initially be marked as “not accepted”. Once the project sponsors have reviewed the application for a particular requirement, if that requirement has been successfully implemented, then that requirement will be marked as “accepted”. This process will also be followed by the development team for earlier testing; however, features are not officially accepted until noted by the sponsors.

Acceptance testing will be considered complete once all requirements with a “High” or “Medium” priority in our Requirements Document have been marked as “accepted”.

Load and Stress Testing
The exact pass/fail criteria for load and stress testing is to be determined by ITS. However, in a general sense, if the application performs to an acceptable degree after it has been put under a certain capacity, then the test item will pass. On the contrary, the test item will fail if the application underperforms.

6 Test Deliverables
The following items are to be delivered as part of this plan:

Acceptance Testing Materials
A spreadsheet will be used to keep track of acceptance test pass/fail criteria. This spreadsheet will track test number, associated user story or use case, pre-conditions for the test, steps to complete the task, the expected outcome, and the actual outcome for all rounds of testing. The actual outcomes will be color-coded based on the feedback provided to show the pass/fail rate at a glance. A second table will contain the metrics on the pass rate.

Usability Testing Materials
The usability testing materials consist of a form containing all the SUS score questions as well as a few general follow-up questions, a spreadsheet of the results, and a spreadsheet with the results of all other metrics tracked during testing and tester thoughts. The team will also create a list of tasks to be tested for each user role to be used during the actual testing.

Screen Prototypes
The screen prototypes can be found on our team website. They were created using Lucidchart. Both clickable and static variations exist for most screens. The screen prototypes are being delivered as part of the test plan as they are required for usability testing, and they may aid in the verification of acceptance tests as well.

7 Environmental Needs
The following elements are required to support the overall testing effort at all levels within the project:

Test Data

Test data will be provided to the development team by the project sponsors in the form of text files. This test data will be designed to work with the old Co-op Evaluation System, and we understand that the format of this data will have to be modified to comply with our new system. Although there are no specific ranges of data that must be provided, we hope to acquire test data for each of our database tables so that we have a better understanding of what type of data the new system will have to work with as we build it.

During the collection of this test data, we understand that the test data we will be given is sensitive information, and is not to exist on our local machines for extended periods of time.

8 Responsibilities

	Dev Team	ITS	OCSCE
Overall testing approach for this level of plan	X	X	
Selection of features to be tested/not tested	X		
Acceptance test documentation	X		
Acceptance test execution	X	X	X
Unit test creation and execution	X		
Usability test creation and execution	X		
Load and/or stress test creation and execution		X	

9 Schedule

A detailed schedule of the entire project is located in our Project Plan; however, this section describes our overall scheduling approach for testing.

Usability testing will take place in the first two to three weeks of the Spring Semester using clickable mockups. Further usability testing will take place throughout the development process as we complete each user role or major milestone. Ideally, this user testing will be run the week after the completion of a major milestone, but we recognize that this may not happen due to our personal schedules and those of our testees. Any delays with this form of testing should not stop development from happening; however, it may result in some amount of rework, especially on the front-end, if significant usability issues are uncovered.

With each cycle, unit and acceptance tests will be delivered that correspond to the functionality set to be completed. Any slippage in meeting this schedule will cause delays into the future cycles as we will have more tests to write in order to reach our minimum of 70% coverage by the end of the project.

10 Planning Risks and Contingencies

Refer to our Risk Table for overall risks to the project, including those with an emphasis on the testing process.

11 References

[1] L. Eder. (2014, June 26). *Stop Unit Testing Database Code* [Online]. Available: http://blog.jooq.org/2014/06/26/stop-unit-testing-database-code

12 Glossary

Term	**Definition**
ITS	Information and Technology Services
OCSCE	Office of Career Services and Cooperative Education
RIT	Rochester Institute of Technology
SUS	System Usability Scale