Project Plan

Co-op Evaluation System

Senior Project 2014-2015

Team Members:

Wayne Starr

Christopher Enoch

William Spaw

Kai Ho

Faculty Coach:

Yasmine Elglaly

Project Sponsors:

Jim Bondi (OCECS)

Table of Contents

Revision History

Version	Edited By	Reason	Date
v1.0	Wayne Starr	Initial Revision	09/15/2015

1. Introduction

1.1 Overview

The purpose of this senior project is to provide a functional Coop Evaluation System that is based on newer web technologies and builds off of work done by a previous senior project team. This project shall future-proof the current system which was created in 2003, and should provide better performance and user interaction. It should also cut down on the random errors that the current system produces, and should be more reliable in general. Most of the changes will be made to the user interface, whereas the existing database should remain similar so as to not interrupt any backend workflows for the system.

As our main goal is to provide a *functional*, *future-proofed* system, priority will be given to extensibility and testing. We will also focus on the core functionality that is needed for the system, with most improvements coming to the emailing and reporting aspects of the previous senior project's code. This will enable us to perform the extra polishing necessary to move this enterprise-level system into production.

1.2 Deliverables

The following deliverables are to be completed by the end of the project. This list is made up of a combination of Software Engineering senior project deliverables and Project Sponsor deliverables.

Deliverable	Description
Project Website	The website which holds deliverables and documents from the project
4-Up Charts	A weekly chart that describes what is done, ongoing, and needed, as well as containing a short list of the risks for that week.
Software Requirements Specification	The document that outlines all of the requirements for the software product that is to be developed
Architecture Specification	The document that outlines the architecture for the software product that is to be developed
Design Specification	The document that outlines the design of the subsystems and services that will make up the software product.
Testing Plan	The plan which outlines what testing framework we will use, what needs to be tested, and how much testing is to be done.

Project Presentations	The presentations to be given to the SE department which will provide status updates on the project. One will be provide an interim status in December, and the other a final status in May.
Project Poster	A poster that describes the project so as to demonstrate the accomplishments of this senior project team.
Technical Report	A report which provides a basic overview of the requirements, and an overview of the system as it stands on delivery.
Support and Maintenance Documentation	Documentation that will allow ITS, EWA, and future teams to support and maintain the system appropriately.
Final Software	The software products for this project including the build scripts and tests

1.3 Assumptions and Constraints

This product must be developed using the ITS supported technologies and frameworks so that it can be maintained and supported in the future. This requires the product to use a Java based framework as the backend, with a recommendation to use Angular.js for the front-end. For the database, we must use Oracle SQL Developer or build off of the structures provided in the original coop evaluation system.

1.4 Definitions and Initialisms

1. ITS - (Information Technology Services)
   a. *A division of RIT that supports RIT's information technology infrastructure*
2. EWA - (Enterprise Web Applications)
   a. *A division of ITS responsible for RIT's high-level web applications.*
3. OCEC - (Office of Cooperative Education and Career Services)
   a. *The office within RIT responsible for managing its Cooperative Education program.*

1.5 Previous Project Materials

The following items were provided to us from previous projects:

Material	Description
ITS Wiki	The ITS wiki describing guidelines for the project as well as information on where

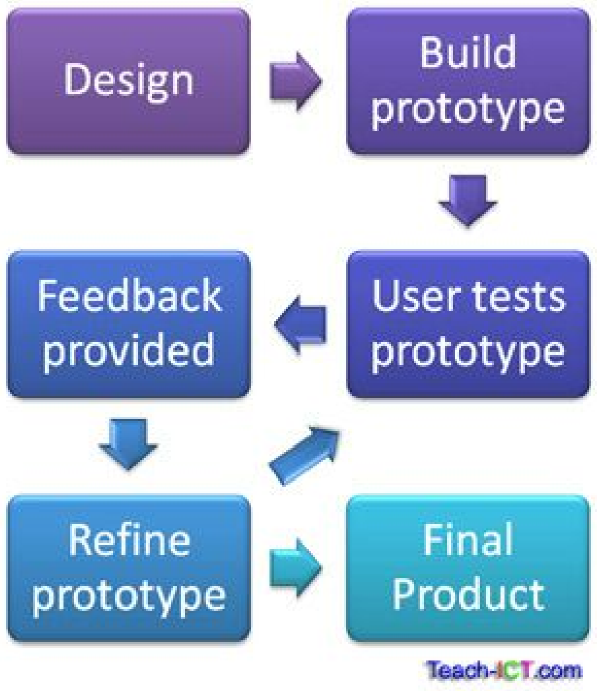
	certain items are in RIT’s web environment.
Project Description	The description of the project provided by the Software Engineering department.
Previous Code Base	The code base from the previous senior project.
Previous Project Deliverable Website	The senior project website and associated deliverables from the previous senior project.

2. Management Structure

2.1 Life Cycle Model

The project lifecycle model that we have chosen for this project is evolutionary prototyping. We have chosen this because of our position as a continuation of a previous team's effort and to ensure that we meet our sponsor's needs as closely as possible. We plan on using the core of the project, already developed by the previous team, and use this as a base to develop a version better meeting our elicited requirements, elicit feedback on the more developed version, and use the feedback to refine the project.

An additional component to the evolutionary prototyping cycle is the reevaluation of project risk after each completed testing and refinement phase.



Because the majority of the design and the initial building of the prototype have already been completed by the previous team, our Design phase is limited to updating the older requirements against current sponsor expectations. Our Build Prototype phase will be similarly limited, it will essentially be a Refine prototype phase, built on the already completed prototype.

Our refine prototype phase will consist of the following tasks:

- Identify target functionality to complete for the cycle
- Reassess and update risks
- Estimate development time and assign tasks
- Develop identified functionality, artifacts, etc.
- Test developed functionality

Our User tests prototype phase will be completed each week by a demo to our project sponsors, along with updating our DEV and TEST servers with the tasks completed in that week's phase. The feedback provided will then be added to the requirements and target functionality for the following phases, time permitting.

2.2 Project Organization

While there will certainly be overlap between roles, the following table lists the major responsibilities of each team member.

Name	Role	Responsibilities
Christopher Enoch	Team Coordinator/Project Manager	● Manages the rest of the coordinators (Activity Tracker) ● Manages Tasks ● Manages Scheduling ● Manages the Trello ● Manages Risk ● Manages the Meeting
Kai Ho	Development Coordinator	● Manages Development Configurations (GitLab, Server, etc.) ● Manages Design Decisions
William Spaw	Scribe/ Sponsor Communicator/Documentation Coordinator / WebMaster	● Manages the Team Site ● Manages the Google Docs ● Manages Minutes / Agendas ● Sends out meeting minutes and agendas, "assigns" parts of agenda
Wayne Starr	Testing Coordinator	● Manages Test plan

		● Gathers User Feedback ● Manages Bug/Issue Tracking ● Executes UAT/Unit Test plans

2.3 Risk Management

We maintain an informal list of week-to-week risks as part of our weekly four-up chart. There is also a separate risk management table where we maintain all risks we identify throughout the project lifecycle, along with their mitigation plans.

The risk management table will be updated each week during the initial planning part of the weekly development phase, detailed in section 2.1. The risk management table includes probable indicators and possible risk mitigation strategies for the most likely risks.

3. Planning and Control

3.1 Estimates

Estimations will be revisited and changed if necessary at the beginning of each phase. We have decided to estimate our phases out until the December break. This will allow us to perform a feature freeze before the break, and spend a couple of weeks ensuring that we maintain a quality system. We will create new estimations for the Spring and add them when they are complete.

Milestone	**Estimated Date of Completion**	**Refinement process**
Requirements	September 24, 2015	The team will analyze what work has been done and the quality of the work, then re-estimate based on their conclusions.
Phase #0 Testing Current System	October 1, 2015	The team will analyze what work has to be done, then re-estimate based on their conclusions.
Phase #1 System Architecture	October 8, 2015	The team will analyze what work has to be done, then re-estimate based on their conclusions.

Phase #2	October 15, 2015	This is a hard deadline. The team will re-estimate a time frame if this deadline is slipped.
Phase #3	October 22, 2015	This is a hard deadline. The team will re-estimate a time frame if this deadline is slipped.
Phase #4	October 29, 2015	This is a hard deadline. The team will re-estimate a time frame if this deadline is slipped.
Phase #5	November 5, 2015	This is a hard deadline. The team will re-estimate a time frame if this deadline is slipped.
Phase #6	November 12, 2015	This is a hard deadline. The team will re-estimate a time frame if this deadline is slipped.
Phase #7	November 19, 2015	This is a hard deadline. The team will re-estimate a time frame if this deadline is slipped.
Phase #8	Tentative	This is a hard deadline. The team will re-estimate a time frame if this deadline is slipped.
Feature Freeze	December 8, 2015	This is a hard deadline. This deadline is set by the team to ensure the work we have done is of a good quality, and that all work is done before the Christmas Break
Finalization of Documentation	December 17, 2015	This is a hard deadline. This deadline is set by the team to ensure the work we have done is of a good quality, and that all work is done before the Christmas Break

3.2 Resources

3.2.1 Time

The calendar time available for this project is two academic semesters, or 33 weeks. This timeline does not include institute breaks.

3.2.2 Cost

At this time, there is no monetary cost to the project. Tools used to benefit the team have free alternatives to them.

3.2.3 Materials

ITS has provided the team with a page of EWA Student Development Guidelines on the RIT Wiki. The wiki provides us with expected standards for technology, documentation, and user interface design. ITS also provides the team with a development/test environment to deploy the application to. This environment will be used for the team to verify that the application will run in an environment similar to the production environment, as well as provide a means for stakeholders to test the application for acceptance.

3.3 Resource Allocation

3.3.1 Additional Manpower

The Office of Career Services and Cooperative Education has two coops available for the team to use. The first works full time at 40 hours a week. The second will begin work the 6th or 7th week of the Fall Semester, and will work 20 hours a week. The team will delegate tasks, approved by the project sponsor and team advisors, to the coops depending upon the time the two will be able to dedicate to the project.

3.4 Tracking and Control

As a team, we will track several aspects of the project through various tools such as Trello, GroupMe, Google Drive, Project Website, and GitLab.

Trello will be used to track progress of deliverables. We will follow a standard practice of Going to Do, Doing, and Done to track the completed tasks of the project. We will also use Trello as a means of tracking Software Engineering deliverables as well. Trello provides a calendar stream that allows tasks assigned to be automatically streamed to Google Calendars.

GroupMe is the group communication tool. Team mates will be able to communicate to one another via the Web or Mobile interfaces provided by GroupMe.

The Google Drive will store all of the team’s documents for the project. These documents will be linked to from the team’s project website.

Finally, GitLab contains an issue tracker that allows team members to report any bugs found. We will use this to not only help the team track bugs that are introduced to the system, but it will help track bugs from the previous team and provide data for metric tracking.

3.5 Metrics

The team will be tracking various metrics to help gauge the quality of the project. Metrics will be gathered and recorded at the end of each phase (where applicable) so the team can make improvements to the quality at the start of each phase.

For the quality of the code base, the team will gather the cyclomatic complexity, or the amount of paths the program will take, at the end of each phase. This metric will be the main driver for refactoring the the past phase. Any classes that come back as having an exceptionally high cyclomatic complexity will be looked refactored. The test coverage of the project will also be monitored.The team will maintain at least 90% test coverage for the code base. Analyzing test coverage will ensure that there are tests for the a large portion of the code base. It will also help reduce the amount of bugs introduced into the system. As well as monitoring the test coverage, the team will gather statistics for the number of bugs found and the number of bugs squashed. This will determine if the team is producing quality code as the projects features are completed.

Finally, we will capture the time it takes to complete a task and the amount of time it takes to complete a task to determine the user acceptance of the system. Doing this will tell us whether certain features need to be reworked to be more usable.

4. Technical Process

4.1 Technology

As we are building upon an existing application, most of the technology has already been decided.

4.1.1 Framework and Tooling

Maven is our build and dependency management system for the Java source code. Currently there is no build system for Javascript and client-side assets so we will use Gulp as our build system and Bower for dependency management.

For the Java server-side, we use Spring Framework with Spring Boot to generate RESTful APIs that are consumed by the client-side. Hibernate is the ORM framework that the application uses to interact with a relational database.

On the client-side, AngularJS is the framework used to build a single-page application.

4.1.2 Environment

Since our server-side technology is Java-based, we will mostly use an IDE such as IntelliJ IDEA or Eclipse to develop. The IDEs mentioned also have support for developing in Javascript for the client-side, however a text editor such as Vim or Sublime Text can also be used.

We use Git as our version control. A remote repository is provided by ITS on its own hosted GitLab solution.

4.2 Infrastructure

The application will be deployed on DEV, TEST, and PROD instances, corresponding to the development, staging, and production environment. ITS will provide us access to a Tomcat instance on DEV as the development server.

RIT users such as students and evaluators are expected to be authenticated by RIT Shibboleth. The application has to integrate with Shibboleth to properly authorize those users.

We use a Oracle database provided by ITS as our relational database. A schema has been created by the previous Senior Project team as part of the application. We will have to expand that to accommodate new functionalities that will be in this release.

5. Supporting Plans

5.1 Configuration Management

Configuration Management information can be found on the ITS wiki for this project. Currently, the ITS managed GitLab is the repository that we will be using.

5.2 Testing

More information about how we perform testing can be found in our testing plan. Any tests that we have written will also be hosted on our GitLab repository.

5.3 Deployment

Deployment information is found on our ITS wiki for the project, but does not differ much from the standard ITS procedures.

5.4 Maintenance

Maintenance information can be found in our supporting maintenance documents, as well as any issues listed on our GitLab repository issues page, our Trello board, or the Technical Report.