|  |
| --- |
| Kyle Bond, Brian Rogers, David Siver |
| Teacher Evaluation System Proposal |
| Goals, User Stories and Processes |

|  |
| --- |
| 2/6/2013 |

Contents

[1. Project 2](#_Toc347953565)

[2. Project Goals 2](#_Toc347953566)

[3. Project Scope 2](#_Toc347953567)

[4. Major User Stories 2](#_Toc347953568)

[5. Roles 3](#_Toc347953569)

[6. Processes 3](#_Toc347953570)

# Project

The Teacher Evaluation System (TES) gives schools the ability to allow managers to evaluate teachers and store evaluation data through a web portal. This flexibility will take an existing system which currently uses Microsoft Excel spreadsheets and convert it to a web2py application. In doing so, teachers and managers will have easy access to current and past evaluation data while allowing school administrators to run statistical analysis on evaluations. Administrators will be able to focus the development of their staffs partially based on the statistics this application will provide.

# Project Goals

Our project will provide an all-inclusive web portal that will allow the entire teacher-manager evaluation process to occur online. Teachers will be presented with a self-evaluation form that allows them to rate themselves, whereas managers will be presented with a very similar form to rate the teachers. These two forms will then be compared, during an in-person interview.

Administrators will be able to view individual interviews and statistics for different hierarchical levels within their school or region. These statistics will provide several different views of the data stored within the database and enable administrators to make educated decisions regarding the future professional development of their teacher community.

# Project Scope

The scope of this project is limited to the following four use cases for this release:

* 1. Administrators who want to utilize the application to produce evaluation statistics.
	2. Evaluators who want to evaluate a teacher.
	3. Teachers who want to self-evaluate.
	4. All roles can view historical data at their appropriate levels.

These three use cases dictate the entire scope of this application – any work that is not in direct support of these use cases should be considered “scope creep”.

# Major User Stories

Below is a list of major user stories that will be used in making detailed system requirements.

* As an administrator, I want a content management system, so that I may change the evaluation criteria.
* As an administrator, I would like to run statistical reports against the evaluation data, so that I may make better professional development.
* As an evaluator, I would like to complete evaluations on teachers, so that we can conduct an evaluation interview.
* As an evaluator, I would like to display personal evaluation statistics to the teacher (during the evaluation interview), so that we can produce a high quality improvement plan.
* As a teacher, I would like to perform a self-evaluation, so that I’m prepared for my evaluation interview.
* As a teacher, I would like to log in to the system and view my historical evaluation data, so that I can monitor my performance.

# Roles

The roles attached to each developer are as follows:

Kyle Bond: Project Manager, Business Analyst, Database Administrator

Brian Rogers: UX Designer, Test Unit Manager

David Sivers: Source Control Manager, Task Manager

All developers will implement source code for the application, as well as perform unit testing on the application once we are “code complete”. Backup roles will be picked up by those with roles most similar to that which needs to be filled. For example, Task Manager and Project Manager have similar responsibilities, as does UX Designer and Business Analyst.

# Processes

Group policies will be enacted to keep this project on task. All team members will adhere to the policies to ensure successful and timely completion of tasks, milestones and the project as a whole. The policy is as follows:

1. After the planning phase is complete, the team will hold weekly meetings, also known as sprint planning meetings, to discuss the development activities for the upcoming week. These meetings will preferably be held on Mondays, during the class period. However, if the meeting is not possible during class time, it will be held directly after class from 7:00 until 7:30.
2. There will be a scrum check-in, in lieu of a daily scrum, completed via the Bitbucket wiki. The scrum check-in will occur between 6:00 PM and midnight on Thursdays. The check-in will consist of these three questions:
	1. What have I done since the last meeting?
	2. What will I be working on until the next meeting?
	3. What, if any, roadblocks are keeping me from completing my work?

The answers to these questions will ensure that members of the group are staying on task and help identify any potential issues that need to be addressed before they become a problem.

1. Team members will be assigned tasks to complete during each sprint planning meeting for the upcoming sprint. It is the team member’s responsibility to complete the assigned task. Failure to complete an assigned task does not relinquish a member from receiving new tasks at the next sprint planning meeting. If a team member feels that they will be unable to complete the task for any reason, they should let their teammates know no later than the scrum check-in on Thursdays.
2. Team members who miss their tasks and do not notify the other team members during the scrum check-in will be required to check in with either Kyle or Brian on a daily basis. This check-in will encourage the delinquent team member to either successfully complete their task or identify roadblocks that the team may need to resolve.
3. Source code management will be handled by David. Checking in of source code should happen in the following circumstances:
	1. A user story/task has been completed.
	2. The application compiles and other tasks are not affected by your current work; however, you are finished developing for the time being.
	3. You are about to make a drastic change to the application and would hate to see it all go down in flames.

Source code should not be checked in, ever, under the following circumstances:

1. The application does not compile.
2. Your source code conflicts with another team member’s code and you are unsure if/how you should merge.