EvaluRate

The Hack Jacks

Dylan Grayson
Conner Swann
Brandon Paree
Brian Saganey

Final Report

May 12, 2016

V1.0

Table of Contents

Introduction

In today's workplace, it's almost impossible to avoid working with a team of people on a project, but there is no standard way for the managers of those teams to evaluate the performance of each individual. Anonymous peer evaluations, forms that every member of a team receives for the purpose of rating their teammates according to some scoring metric, could solve that problem. However, there is no easy way to create and distribute these peer evaluations. EvaluRate aims to provide a robust software platform for distributing anonymous peer evaluations and viewing the resulting data. EvaluRate also allows for the modelling of real world division hierarchies, so virtually any manager at any level in any organization could issue anonymous peer evaluations to a team of subordinates.

EvaluRate will mainly be used for corporate and academic environments.

- Corporate environments will use the anonymous peer evaluations to inform staffing and human resource decisions, as well as better inform future team roster decisions.
- Academic environments will use the anonymous peer evaluations to score each individual who contributed to a group project, thus ultimately determining their grade for the project.

Since EvaluRate is a fairly complex application for administrative roles, the risk was of the user interface being too complicated for administrative users. The design process proved difficult, yet helped refine the user interface to the point where it can be used and understood easily. This was partly because the application was built with Meteor Javascript framework, which provides easy tools to build a reactive web application with interactive client-side web pages. The end result is a web application that is both robust and elegant.

Process Overview

The process of developing EvaluRate was an agile methodology similar to Kanban. It relied heavily on the use of Github, and the issues system therein. Basically, every task that needed to get completed was created as an issue on Github and briefly described. As developers, we would assign ourselves an issue, complete the issue, then create a pull request for that issue. The pull request is then assessed by one or more teammates and, if everything works, merged into master. The corresponding issue is then closed.

For the purpose of assigning roles to each developer, we must first clarify that the Meteor paradigm includes a robust client application, therefore much of the business logic is contained in the client application. Since it is more common in web development for the majority of business logic to exist on the back-end/server, we use the terms like this:

- *back-end* - The business logic of the application whether it exists on the client or the server.
- *front-end* - The actual user interface on the client, and other visual components of the application.

Roles for Each Member

- Dylan Grayson - Team leader, back-end developer.
- Brandon Paree - Back-end and front-end developer.
- Conner Swann - Back-end developer.
- Brian Saganey - Front-end developer.

Requirements

This section contains a brief overview of the most important aspects of the requirements for EvaluRate.

Functional Requirements

1. Unit Management
    1.1. User can create new units
    1.2. User can administer and set permissions for units
    1.3. User can view units they belong to
2. Project Creation
    2.1. User can create a project with the following criteria
        2.1.1. Name
        2.1.2. Start date
        2.1.3. End date
        2.1.4. Team list
            2.1.4.1. Complete with team roster
3. Evaluation Engine
    3.1. Issue evaluations to a project
        3.1.1. Sent to each member of each team in the project
        3.1.2. Each member sees an evaluation pertaining to their team
    3.2. Submit Evaluation
        3.2.1. User can fill out peer evaluation
        3.2.2. User can submit peer evaluation

Non-functional Requirements

1. Works with any organizational structure
    1.1. Can define any organization to map to a real world organization with people and units.
2. Real time updates

2.1. All changes to the database are reflected on client application immediately

Architecture

The high level architecture (shown in Figure 1) is quite simple. The basic organizational structure is the unit which can contain people, projects, or more units. Projects contain teams (which are technically just temporary units) and evaluations. Evaluations are attached to projects and are sent to every member of every team. The majority of this logic is implemented in a JavaScript client-side application, which consumes only certain data from a minimal server-side JavaScript application in a Node runtime.

Initially, projects were not part of the system, as we planned to allow the arbitrary creation of units as teams, and the ability to attach an evaluation to the unit containing the teams. We eventually decided to use projects to act as a container for teams and evaluations, because it is a stronger metaphor to reality.

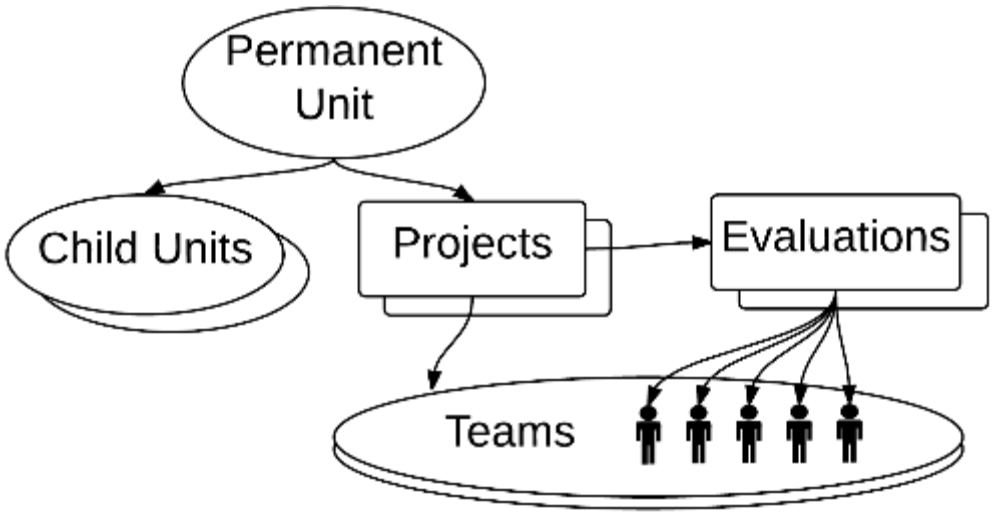

Figure 1 - Architectural diagram

Testing

EvaluRate is a fairly complex web application that aggregates data from multiple different users, and has a strict permissions hierarchy. It also relies on the usability of a front end application that faces two distinct user groups. For these reasons, our primary focus for testing EvaluRate will be on:

- Unit testing - Testing for code correctness
- Usability testing - Testing for usability from target users

Unit Testing

Since EvaluRate is built with Meteor.JS, all unit testing is performed by the node-based unit testing framework Mocha. Using the framework, we will be testing EvaluRate in two major ways. At the method-level, we will be performing simple unit testing. However, due to the closely integrated nature of the client and server, simple unit testing does not adequately capture the range of possible component interactions. In other words, testing only on the server or only the client is insufficient, so Meteor provides a testing environment that lets us load both client and server code to their respective areas while also isolating those components from the rest of the application. The two main components of the application that were tested are as follows:

- Result Scores - The weighted average of evaluation data for each member on a team.
- Permissions - The restriction and denial of different administrative actions on any given unit.

Usability Testing

In order to decide how the usability for EvaluRate should be tested, we first had to consider who would be using the application. There are two obvious divisions in the functions of the user interface. The first function facilitates administrators/managers of

projects/teams to issue evaluations to the members in those teams, and receive the data for viewing. The second function allows the members of those teams to fill out and submit the evaluations. The usability testing for EvaluRate included user feedback, and informal walkthroughs from two user groups we have easy access to:

- Professors: They fit right in our user group for project/team administrators, as they often assign group projects, and may desire peer evaluations.
- Students: They are the user group that will be put into teams on projects, therefore taking evaluations.

These professors provided significant feedback that exposed some weak points in the user interface where the appropriate steps were not clear.

Results of Testing

Due to time constraints, the testing occurred much later than expected, and therefore we only had time to change a few minor things:

- Helpful tooltips on buttons with only icons
- Confirm password on sign up

Future Work

Dr. Doerry is actively looking for a Computer Science student at Northern Arizona University to continue development and maintain EvaluRate. The project can be improved by adding more evaluation criteria and refining the user interface.

It is our hope EvaluRate will become a valuable tool for the Computer Science department for peer evaluations, and EvaluRate will be worked on by future students in the Computer Science department.