Improving Business Development Using Gamification

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Marcus Mann

May 8, 2023

Capstone advisor: Rosanne Vrugtman, Department of Computer Science

Improving Business Development Using Gamification

CS4991 Capstone Report, 2023

Marcus Mann
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
mpm7cf@virginia.edu

ABSTRACT
Elder Research, a Charlottesville-based data consulting company, needed to improve their understanding of employee knowledge regarding industry tools and methods, which required individual employees to input their experience into the internal company wiki, "The Vault". To encourage employees to input their data, I designed a point and badge system, which rewarded users for contributing to The Vault and completing technical certifications. I achieved this using the Ruby on Rails framework, PostgreSQL, and expanding Redmine's functionality. For the point system, I designed a leaderboard system for users to compare their point totals, and for the badge system, I used design philosophies from 2007 video game Halo 3. As a result, Elder Research's business development team was able to encourage the 50+ technical employees to adequately populate The Vault with employee knowledge. Additionally, the gamification system encouraged employees to compete for a higher position on the point leaderboard. In the future, I plan to account for length of content edited to encourage detailed contributions.

1. INTRODUCTION

How can a data consulting company ensure that their business leaders have an accurate representation of their employees' knowledge base across different technology stacks? To address this problem, Elder Research needed to create an internal system that allowed employees to share their knowledge quickly and easily. However, anecdotally, employees do not enjoy maintaining a wiki with their work experience. To intrinsically motivate the employees, I designed a point, leaderboard, and badge system that rewarded users for contributing to the company's internal wiki. The Vault system was created using the Ruby on Rails framework and PostgreSQL, and by expanding the functionality of Redmine, an open-source issue tracker Elder Research leveraged to organize company knowledge. Through this system, Elder Research was able to encourage employees to share their knowledge and stay up to date on their industry's tools and methods.

2. RELATED WORKS

The most influential related work was from the video game Halo 3 (2007). In Halo 3, medals would be awarded to the player for completing specific objectives, such as getting five kills in a row (Fig. 1). However, I found little research regarding specific implementations of gamification systems.

One system I investigated was the Stack Overflow reputation system (Stack Overflow, n.d.), which grants users reputation based on feedback from other users. Even though the presentation of the reputation matched the goal for Elder Research's system, the fundamental goal was different. Stack Overflow says their reputation system "is a rough measurement of how much the community trusts you," which is not applicable to the Vault, given that we can trust all employees to update the Vault with accurate information.

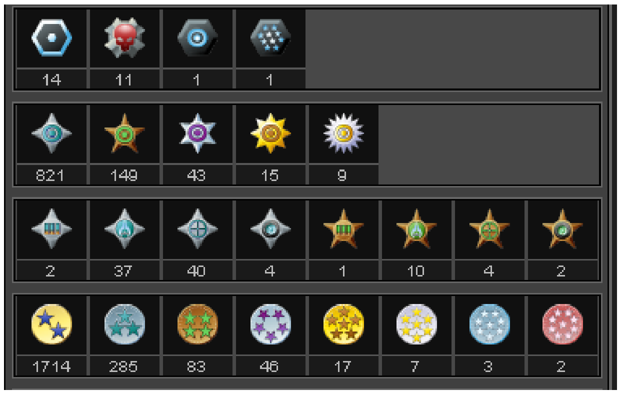

Figure 1—Halo 3 Medical Chest

3. PROJECT DESIGN

The project required modifications of the existing Redmine system to capture and display the points and badges. These were broken down into several different features.

3.1 SYSTEM REQUIREMENTS

In order to meet the client's end goal and make a product that is designed to work as they intend, it is important to acquire a comprehensive list of system requirements.

3.1.1 Point System

Minimum Requirements

As a user, I should be able to:

- receive a point if I make an edit on a wiki page in The Vault
- compare the amount of points I have accrued to other users on a leaderboard
- view the number of points I have accrued on any page of The Vault.

3.1.2 Badge System

As a user, I should be able to:

- view the number of badges I currently have on any page of The Vault.
- view all of the badges that I could earn.
- view who has earned a badge.
- view the cumulative number of badges all users own.
- nominate another user for a badge.

As an administrator, I should be able to:

- approve a badge nomination.
- submit a new type of badge, including the name of the badge and the image of the badge.
- approve a new type of badge.

Desired Requirements:

As a user, I should be able to:

- view badges in a similar manner to Figure 1.

3.2 SYSTEM LIMITATIONS

Because The Vault is a Redmine- (and therefore Ruby on Rails)-based web application, the point system and badge system both needed to be designed in accordance with Redmine design patterns. These closely matched Ruby on Rails Model-View-Controller (MVC) patterns, meaning we were required to use ActiveRecord for interacting with PostgreSQL for designing our data models, and using the Embedded Ruby (ERB) templating system. Redmine introduced additional limitations for recommended design patterns: Redmine introduced the notation of "`acts_as_attachable`," which is a model-agnostic way of attaching files from the file system onto a data model.

Redmine also introduced the notion of plugins and hooks, which are used to add more features to the codebase without overwriting code from the original codebase. I found this essential for reducing the difference between the base Redmine codebase and our modified Vault codebase. I found it important to keep them similar because Redmine regularly released security and feature updates, and if we overwrote base Redmine code, it created Git merge conflicts in the future.

3.3 KEY COMPONENTS

We created different specifications for the point and badge systems, which mostly revolved around the database models. We also ran into Redmine-specific challenges, such as sparse documentation and deployment challenges.

3.3.1 Specifications

For the point system, I originally considered a system where we would add a database migration that would expand the User model to record a point value. However, this system was not dynamic enough for the client's needs, as they wanted to deterministically recreate the point value of each user at any point in time. Fortunately, Redmine's wiki system kept track of each change a user made to each wiki page, meaning that summing the number of wiki edits a user made was possible. Because of this, we were able to create a SQL view that aggregated user contributions as a single number value. This satisfied the client requirements of a deterministic, dynamic point calculation.

For the badge system, three models were required: a Badge model, a Badge Group model and a Nomination model. The Badge model contained an author, a BadgeGroup, a user group, a many-to-many relationship with the Nomination model, a many-to-many relationship with the User model, and an image, which was attached with the "acts_as_attachable" extension provided by Redmine.

The BadgeGroup model was responsible for organizing the visualization of the badges to match Figure 1. In the end, we had a name, description, and priority field for badge groups. The priority field was responsible for allowing the users to control the order of the rows of badges in the interface.

Finally, the Nomination model was responsible for tracking the process of giving a badge to a user, from nomination to approval/rejection. A nomination represented the many-to-many relationship between badges and users. We used the "journal" system, in which each change would be stored as a new row in a database table, and the aggregate of changes would represent the current version of a Nomination. This journal system was used in the wiki page model in the Redmine codebase and was adapted for my use case. A nomination would have 3 states: nominated, granted, and rejected.

3.3.2 Challenges

We encountered multiple challenges. Ruby on Rails has a plethora of documentation, but Redmine's documentation was considerably sparser. This made interacting with the attachment system and the journal system considerably more tedious than necessary.

However, the largest challenge was a lack of hooks for introducing custom behavior into base Redmine interfaces. While adding menus for the badge display page was well supported, meeting the client requirement of "As a user, I should be able to view the amount of points/number of badges I currently have on any page of The Vault" required editing base Redmine code, which introduced necessary technical debt. Additionally, the client deployed Redmine with JRuby, a Ruby interpreter written in Java, which was no longer supported by Redmine. Finally, the vast test suite provided by Redmine relied on the default localization files, which the client overrode with company-specific language. For instance, in a default version of Redmine, each ticket would be described as an "Issue." However, in The

Vault, this was renamed an “Artifact.” The test suite written for Redmine assumed this when testing, resulting in the entire Redmine suite not being available. This led to maintenance issues that have yet to be resolved.

3.3.3 Solutions
While the temporary solution for minimum viable product (MVP), the JRuby issue has been resolved by switching to a Docker based deployment, allowing us to mitigate the Windows-specific issues of CRuby, the default Ruby interpreter, not being compatible with the operating system. Instead, by running in a Linux-based Docker container, we can control our environment to use the correct PostgreSQL drivers, install the correct dependencies, and successfully deploy regardless of which system we are running on.

4. RESULTS
As of March, 2023, users have nominated each other for over 50 badges, used the badge system to track certifications, and overall users have accrued over 2000 points across <200 employees. These certifications are in the process of being integrated with the client’s resume generating system, which would be used to generate employee resumes to bid on contracts in the future.

5. CONCLUSION
This project has successfully addressed the need for an efficient and engaging method of capturing and maintaining employees' knowledge within Elder Research. The implementation of a point, leaderboard, and badge system has proven to be a meaningful and valuable approach, resulting in increased employee engagement and better knowledge sharing across the organization. By combining gamification elements with practical applications, such as tracking certifications for the company's resume-generating system, the project has not only enhanced the internal knowledge base but has also provided tangible benefits for clients and future business opportunities. As the company continues to grow and evolve, this innovative solution will remain a key asset in ensuring a comprehensive and up-to-date understanding of industry tools and methods among employees.

6. FUTURE WORK
Elder Research's current implementation of the point, leaderboard, and badge system has been successful in promoting employees' contribution to the internal wiki. However, there is still an opportunity for improvement. One suggestion is to integrate content length as a factor in determining rewards; this incentivizes staff to produce more substantive contributions that benefit the company's knowledge base. Presently, the system rewards points based on how many edits were made without considering the quality or length of the revision. Enhancing the point reward system will encourage employees to provide higher quality contributions.

REFERENCES
Stack Overflow. (n.d.). *What is reputation? How do I earn (and lose) it?* Stack Overflow. https://stackoverflow.com/help/whats-reputation

Halo 3. Xbox 360, Microsoft Game Studios, 2007.