Gamification of Data

***Test Requirements Document***

**Senior Design - CIS 4952 - Winter 2020**

**Version No. 1.1**

Team Members:

Cameron Kozan

Zaid Alsafi

Alex Navarre

**Project Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version Number | Date | Revising Author | Description of Revision |
| 1.0 | 3/15/2020 | Alex Navarre | Document Created |
| 1.1 | 3/16/2020 | Zaid Alsafi, Cam Kozan, Alex Navarre | Updated |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| **Table of Contents** |

[**1.0 Introduction**](#_83ik91ktkl1p) **4**

[1.1 Goals and Objectives](#_orgkxpxhw7vb) 4

[1.2 Statement of Scope](#_ungjh4hatje0) 4

[1.3 Major Constraints](#_he46pyfvih) 4

[**2.0 Testing Plan**](#_6r77i7146eai) **4**

[2.1 Software (SCIs) To Be Tested.](#_603luj9271tf) 4

[2.1.1 Interfaces](#_25c4owxtoafa) 4

[2.2 Testing Strategy](#_cb7o4hmg8for) 6

[2.2.1 Unit Testing](#_14v2uswi8a1x) 6

[2.2.2 Integration Testing](#_wq3ka5p5g8ko) 6

[2.2.3 Validation Testing](#_onlfgnxokv5d) 6

[2.2.4 High-order Testing](#_dh5pgf51j03o) 7

[**2.3 Testing Resources and Staffing**](#_o2vi9lrt3imc) **7**

[2.4 Test Record Keeping](#_fgr9jnoaqbk6) 7

[2.5 Testing Tools and Environment](#_ag508py9hbrg) 7

[2.6 Test Schedule](#_6k80chnfq4gz) 7

[**3.0 Test Procedure**](#_mftvunn53dp6) **8**

[3.1 Software (SCIs) To Be Tested](#_9zce3jy0f4eb) 8

[3.2 Testing Procedure](#_j5542z2ibruj) 8

[3.2.1 Unit Testing](#_dcfwriw2u3kz) 8

[3.2.2 Integration Testing](#_c874w9bv03if) 9

[3.2.3 Validation Testing](#_yirmvn5rn9xy) 10

[3.2.4 High-order Testing](#_wad0lycoyrz) 10

[3.3 Testing Resources and Staffing](#_1tb0m12mzs0a) 10

[3.4 Test Record Keeping and Log](#_kniy5xy2768g) 11

# 1.0 Introduction

## 1.1 Goals and Objectives

The goal of this project is to develop a gamified web application. The gamified web application will allow for the workers to be more motivated along with increasing productivity. The end goal of this project is for a user to go into the web application see how they rank among their peers, look at achievements, along with how all their teams are doing as well.

Our goal is to deliver a program that has no errors or bugs. We will be working together with our development team and our customer to ensure that our program is being built as close to their specifications as possible. If any unexpected errors do arise, we will include them in the Test Requirements Document and notify our customer as soon as possible.

## 1.2 Statement of Scope

The Gamification software will use the client’s API, Invisitag, which will allow us to see all of the teams’ assets. From their api, each asset contains a cost and a weight, which we have incorporated in our website. Our website will allow admins to edit the cost and the weight of each asset. The employees will each have a score per every time they check in their assets. This score is comprised of the asset weight multiplied by if it’s stolen, lost or it’s their own asset. From there, the user will see a dashboard with all of the statistics, such as how they rank among their teammates, their score over time graph, along with many more metrics. The admins will assign employees to their respective teams, which then the employee’s performance will be based on the team performance.

## 1.3 Major Constraints

* Web application front end, back end and service must all run on EC2
* Database must be hosted on AWS RDS
* Must be done by April 2020
* In person/online meeting

# 2.0 Testing Plan

Our goal is for the software to be bug free. We want to spend enough time testing the program to try and avoid any defects. A majority of our remaining time is going to be spent testing. Below is the description of the testing procedure and strategy.

## 2.1 Software (SCIs) To Be Tested.

### 2.1.1 Interfaces

* Login
  + The login will be tested with multiple accounts. Multiple administrator and employee accounts will be created and tested to ensure proper results.
* Dashboard: Administrator & Team
  + The dashboards will be tested for regular employee and administrator accounts. After logging in, every user will land on the Dashboard page. Depending on the user type, either the team or administrator dashboard should appear. Both account types will be used in testing.
    - Current Rank should be retrieved from the database, calculated, and displayed properly
    - Tools Lost should be retrieved from the database and displayed properly
    - Badges should be retrieved from the database and displayed properly
    - Rank Change should be retrieved from the database and displayed properly
    - The Leaderboard should be pulled from the database and displayed in the correct format.
    - The Average Score chart should retrieve information from the database
* View Profile
  + Experience, Recent Activity, Challenges, Badges, and Item Details should all be:
    - Retrieved from the database
    - Calculated (if necessary)
    - Displayed properly
* Manage Team
  + Teams and Activity should be:
    - Retrieved from the database
    - Displayed properly
  + Add Teams button should display a list of all possible imported teams to add to the full Manage Teams list.
    - The ‘Save’ button on the Import Teams menu should update the database and reload the Manage Teams page with the imported teams added to the list.
  + The ‘Edit’ button should navigate the user to the Team Detail page.
* Team Detail
  + The ‘Add’ button should open the Add Members menu
    - The Add Members menu should:
      * Retrieve all employees from the database
      * Display First & Last Names, Email, and Images of all employees not currently assigned to the selected team.
    - The ‘Save’ button should update the database and apply changes to the Team Detail page.
* Manage Employee
  + First Name, Last Name, Email, Image, and Activity should all be:
    - Retrieved from the database
    - Displayed properly
  + For an administrator account, the ‘Add’ button should open the Create Employee menu.
    - The Create Employee menu should include the following fields that are stored and retrieved from the database:
      * First Name
      * Last Name
      * Email
      * Start Date
    - The ‘Save’ button should update the database and apply changes to the Manage Employee page.
* Employee Detail
  + First Name, Last Name, Email, Image, Activity, and Teams should all be:
    - Retrieved from the database
    - Displayed properly
  + The ‘Add’ button should open the Assign Employee to Teams menu
    - The Assign Employee to Teams menu should retrieve all teams from the database and display them in alphanumeric order
    - The ‘Save’ button should update the database and add the employee to the selected teams.
  + The ‘Remove’ button should update the database and remove the employee from the selected team.
  + The ‘Deactivate’ button should update the database and deactivate the selected account.
* Manage Assets
  + Asset, RFID, Cost, and Weight should all be:
    - Retrieved from the database
    - Calculated (if necessary)
    - Displayed properly
  + The ‘Add’ button should open the Import Tags menu
    - The Import Tags menu should load all the current importable assets from the database and list them in alphanumeric order.
    - The ‘Select All’ button should select all items in the list
    - The ‘Save’ button should update the database and add all the selected imported asset tags to the Manage Assets list.
  + The ‘Edit’ button should navigate the user to the Edit Assets page.
* Edit Assets
  + The Current Values should be retrieved from the database and displayed properly.
  + When saved, the Modified Values (Cost & Weight) should be updated in the database and the Manage Assets page list should be updated.

## 2.2 Testing Strategy

### 2.2.1 Unit Testing

Our team will be unit testing each component individually. We will be using both black and white box testing methodology. Each component will be tested by grabbing data from the database and we will be checking the output. We will be doing this to find any bugs that the unit of code will have.

### 2.2.2 Integration Testing

Each member will be performing testing the individual component before the unit is integrated. When all team members agree that an individual component can be integrated that code will be sent to the main branch on github. When the integration is completed two members will go through a checklist and see if anything unexpectedly changed it will be logged and what severity the bug will be. If the severity is extremely high both members will try to find the issue and fix the problem before integrating any unit.

### 2.2.3 Validation Testing

For validation testing, we will be testing with the client on live servers to ensure the website matches the clients requirements. We will go through every feature with them to make sure it meets their expectations. Black box testing will be used for our validation testing with our customers.

The database is hosted on AWS RDS and the front and backend programs are hosted on AWS EC2 instances.

### 2.2.4 High-order Testing

For high-order testing we will be using different test methods to test the various conditions:

* Stress Testing
  + We will be testing at various frequencies with randomly generated data inputs, in large sums, to validate the braking point of each individual interface.
* Performance Testing
  + We will be testing the range of performance to gauge the softwares compatibility and functionality. We will be mainly testing the Manage Asset page to judge the loading time of all the assets from the database. This will let us confirm that the minimum loading times are within the customer specifications.
* Security Testing
  + We will be using SQL injection testing and password salting to judge the level of security being provided.

## 2.3 Testing Resources and Staffing

Resources:

* Windows/Mac/Linux Computer
* Internet Connection
* Internet Browser
* Working AWS Servers

Staffing:

* All members will participate in testing the software.

## 2.4 Test Record Keeping

Test record keeping and Test Work Products are described in section 3.4 of the Test Specifications Document. For information regarding these topics, please refer to section 3.4 of the Test Specification Document.

## 2.5 Testing Tools and Environment

Visual Studio Community will be used to test the backend program in C#. The frontend testing will be done using Visual Studio Code and Google Chrome for the web browser. The inspect function (F12) in Google Chrome will be used for debugging.

All of the sample data is pulled from the Invisitag API. Dummy teams, assets, and employees have all been provided through the API and are updated every night. Ranking data will be determined from the sample data provided and ranks will be calculated at set intervals.

## 2.6 Test Schedule

Below is a detailed schedule for our unit, integration, validation, and high-order testing.

* Unit Testing
  + 03/01/2020 - 03/15/2020
* Integration Testing
  + 03/16/2020 - 03/18/2020
* Validation Testing
  + 03/18/2020 - 04/20/2020
* High-order Testing
  + Stress Testing
    - 03/16/2020
    - 04/06/2020
  + Performance Testing
    - 04/07/2020
  + Security Testing
    - 03/01/2020 - 03/15/2020
    - 04/04/2020

# 3.0 Test Procedure

## 3.1 Software (SCIs) To Be Tested

Please refer to section 2.1 of the Test Specification Document for information on the software to be tested and any exclusions.

## 3.2 Testing Procedure

### 3.2.1 Unit Testing

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Description | Input |
| 1 | Black Box Testing | Testing Login | Username: [cckozan@umich.edu](mailto:cckozan@umich.edu)  Password: Yeet123 |
| 2 | Black Box Testing | Testing if an admin can create a team | Under Manage Teams -> Add team |
| 3 | Black Box Testing | Testing if an admin can delete a team | Under Manage Teams -> Delete team |
| 4 | Black Box Testing | Testing if an admin can add employees to a team | Under Manage Teams -> Add Employee |
| 5 | Black Box Testing | Testing if an admin can remove employees to a team | Under Manage Teams -> Remove Employee |
| 6 | Black Box Testing | Testing if admin can create  employee account. | Under Manage Employees -> Add employee |
| 7 | Black Box Testing | Testing if admin can delete  employee account. | Under Manage Employees -> Delete employee |
| 8 | Black Box Testing | Testing to see if a user gets a badge once they complete a challenge | Give a user data to complete his challenge |
| 9 | Black Box Testing | Testing to see if a user levels up once they hit the required XP to level up | Give a user enough XP to level up |
| 10 | Black Box Testing | Testing to see if a user can see their teams dashboard | Navigate to the Team Dashboard page |
| 11 | Black Box Testing | Testing to see if a user can see their own stats | Navigate to the view-profile page |
| 12 | Black Box Testing | Testing to see if an admin can see all the teams dashboards | Navigate to the admin dashboard and click on each individual team |
| 13 | Black Box Testing | Testing to see if another user can see if the can see other user’s profiles | Navigate to the dashboard, click on another team on the leaderboard |
| 14 | Black Box Testing | Testing to see if a user can reset their password | At the login screen, hit reset password |
| 15 | Black Box Testing | Upload an image | Go to the view-profile page -> manage profile -> upload image -> submit |
| 16 | Black Box Testing | Change a user’s name | Go to the view-profile page -> manage profile -> change their name-> submit |
| 17 | Black Box Testing | Move a user from team to team and see if his score changes | Check the user’s profile, then remove them from team A and add them to team B and see if the score has changed. |
| 18 | Black Box Testing | Add a team to a user and see if his score changes | Check the user’s profile, add them to another team and see if the score has changed. |
| 19 | Black Box Testing | Remove a user from a single user team | Check the user’s profile, then remove them from team A and go back to the profile page and see if their score has changed. |
| 20 | Black Box Testing | Checking to make sure the styling doesn't completely mess up once the width changes | Change the dimensions of the screen and see the changes |

### 3.2.2 Integration Testing

We will be testing the software components individually and locally to begin. We will perform the full software integration testing after all of the individual modules have passed their testing.

Testing will be run locally before being run on AWS servers. This will allow us to test the software anywhere, as well as letting our customers have direct access. They will be able to help provide input and report bugs with the software.

Each of the software (SCIs) items will be the test cases for the integration testing. Each page or menu as a whole will be involved in a test case. We will be testing each and every page and/or menu for all the errors that logically can occur.

At the end of the test, all of the results should be positive. All of the software components and pages should be working as intended. In the case we come across any unexpected errors, we will record them in the Test Requirements Document to be fixed at a later time.

### 3.2.3 Validation Testing

Validation testing will be used to ensure the software does what is required in the customers specifications. We will be using black box testing throughout the entire development process to ensure our program is behaving as intended. We will be working with our customer on a bi-weekly basis to go over our progress and report any unexpected errors.

Our entire team will be participating in our validation testing. Each member will run the designated test cases on their own, locally and online, and then we will compare our results. If we encounter any errors, we will report to each other as soon as possible in order to try and mitigate the issues. Any unresolved issues will be noted in the Test Specification Document and reported to our clients during our bi-weekly meetings.

The software will be validated by our customer after all of our testing has been completed. If there are any unresolved errors or bugs, we will attempt to come to a compromise with the customers and lay out a plan to fix the issue at a later date.

### 3.2.4 High-order Testing

High-order tests are a combination of several different test methods. We will be running three types of high-order tests on our software. These tests will be performed by all the software developers and with our customers to validate the cases.

* Stress Testing
  + We will be testing at various frequencies with randomly generated data inputs, in large sums, to validate the braking point of each individual interface.
* Performance Testing
  + We will be testing the range of performance to gauge the softwares compatibility and functionality. We will be mainly testing the Manage Asset page to judge the loading time of all the assets from the database. This will let us confirm that the minimum loading times are within the customer specifications.
* Security Testing
  + We will be using SQL injection testing and password salting to judge the level of security being provided.

## 3.3 Testing Resources and Staffing

Resources:

* Windows/Mac/Linux Computer
* Internet Connection
* Internet Browser
* Working AWS Servers

Staffing:

* All members will participate in testing the software.

## 3.4 Test Record Keeping and Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Expected Output | Actual Output | Pass/Fail | Date Fixed (or N/A) |
| 1 | User should be logged in | User Logged in with their personal information | Pass | N/A |
| 2 | Admin should be able to pull teams from the database | Admin is able to pull teams that have not been able to be pulled | Pass | N/A |
| 3 | Admin should be able to delete a team | Admin is able to delete any team | Pass | N/A |
| 4 | Admin should be able to added an employee to a team | Admin is able to add an employee to a team | Pass | N/A |
| 5 | Admin should be able to remove an employee from a team | Admin is able to delete an employee from a team | Pass | N/A |
| 6 | Admin ,should be able to create an employee account | Admin is able to create an employee account | Pass | N/A |
| 7 | Admin should be able to delete an employee account | Admin is able to delete an employee account | Pass | N/A |
| 8 | The user should now have the badge for the challenge they completed | N/A | N/A | N/A |
| 9 | The user should now be one level higher | N/A | N/A | N/A |
| 10 | The user should be shown their own teams dashboard | User is shown their team dashboards | Pass | N/A |
| 11 | The user should be shown their own profile (Statistics) page | User is shown their own profile page | Pass | N/A |
| 12 | The admin should be able to access all of the team’s dashboards | The admin has access to all the team’s dashboard | Pass | N/A |
| 13 | The user should be able to see their peer’s profile | The user is able to see their peer’s profile | Pass | N/A |
| 14 | A user should be able to reset their password | A user gets an email with their new password | Pass | N/A |
| 15 | The user should be able to see their newly updated profile picture | User is able to see their newly updated profile picture | Pass | N/A |
| 16 | The user’s name should be changed | N/A | N/A | N/A |
| 17 | The users score should reflect the new team’s score, rather than the old | N/A | N/A | N/A |
| 18 | The users score should reflect the old team + the newly added teams score | N/A | N/A | N/A |
| 19 | The users score should reflect the single team they’re on | The users score does reflect the team they are on | Pass | N/A |
| 20 | The styling should remain good looking under compression | N/A | N/A | N/A |