**Big Data**

**Gamification**

**Test Specification**

Ali Tarraf

Ali Al-mugoter

Steve Karim

[**Introduction**](#_3yhrbwaauo6w) **4**

[Goals and objectives](#_oslz58g3sznh) 4

[Statement of scope](#_muvmue7et2cn) 4

[Major constraints](#_x8u4j4dc134p) 5

[**Test Plan**](#_r2jvofcjmf0) **5**

[Software (SCIís) to be tested](#_oa5a908v14v5) 5

[Testing Strategy](#_veoju53xu1rm) 5

[Unit Testing](#_ci5xu939g1x7) 5

[Integration Testing](#_oe88ugfp28t8) 5

[Validation testing](#_i1dm9mycr6e4) 5

[High-order testing](#_exa00jxxyosa) 5

[Testing resources and staffing](#_fco5hzubf6su) 6

[Test work products](#_m1448zcvi54p) 6

[Test record keeping](#_5yhd46q4pk3z) 6

[Test metrics](#_53gdbp4x0518) 6

[Testing tools and environment](#_gksgiwuu9l7g) 6

[Test schedule](#_nqfaaem31z23) 6

[**Test Procedure**](#_i2zaf7gghjl2) **6**

[Software (SCIís) to be tested](#_9mlxpqj1wg5h) 6

[Testing procedure](#_fgpscfckoaj3) 6

[Unit test cases](#_hpgpsxdwb9bj) 7

[Stubs and/or drivers for component i](#_tn0q5l3tk7rn) 7

[Test cases component i](#_u16jgsft6eco) 7

[Purpose of tests for component i](#_8dym7wd8bfbz) 7

[Expected results for component i](#_pr3sebfk2bsk) 7

[Integration testing](#_yajj3aikujqi) 7

[Testing procedure for integration](#_nkyebhanwomf) 7

[Stubs and drivers required](#_ezqe1qorqdts) 7

[Test cases and their purpose](#_i9neehxg793o) 7

[Expected results](#_2unsrlxhk9py) 7

[Validation testing](#_5y1x1iw8mbkb) 7

[Testing procedure for validation](#_57zui7mtvtoq) 7

[Expected results](#_rs3b3nvkra1s) 7

[Pass/fail criterion for all validation tests](#_v9ijzob7biv1) 7

[High-order testing (a.k.a. System Testing)](#_gkekx89vbgdk) 7

[Recovery testing](#_yxnrdl6wgywe) 8

[Security testing](#_9bc65ewzl7qy) 8

[Stress testing](#_s60wmknhlbwr) 8

[Performance testing](#_i3buqjz2fyuo) 8

[Alpha/beta testing](#_oyf3z6z65fwx) 8

[Pass/fail criterion for all validation tests](#_7yjdcdbf47op) 8

[Testing resources and staffing](#_2c8oc0xma7ma) 8

[Test work products](#_20x5n54wthhm) 8

[Test record keeping and test log](#_kednolxqjhbk) 8

# Introduction

This section provides an overview of the entire test document. This document describes both the test plan and the test procedure.

## Goals and objectives

This project is to develop a web application for the gamification of big data. To ensure the best quality of the software, the development team will carefully review and test all functionalities. Code will be tested before being committed to the master branch. It will then be reviewed by two other developers for approval before being pulled. The software will then be tested again to make sure the new functionality did not break other functionalities. Once the final version of the software is complete the entire program will be tested again to ensure everything is working correctly. Any bugs that are found will be recorded and the development team will decide the best course of action to eradicate those bugs.

## Statement of scope

A description of the scope of software testing is described in this section. Below functionalities, features, or behaviors to be tested are noted. In addition any functionalities, features, or behaviors that will not be tested are also noted.

The scope of the testing will focus on providing a software that functions properly, the essential functionalities have been implemented and are as follows: user login, team management, user management, asset management, rank history, and rankUpdater Service. The main focus of testing is to provide a solid working foundation with these functionality allowing the client be confident in the product they are deploying into a production environment.

* Unit Testing - Using Black Box Testing
  + Web Application
  + Database
* Integration Testing
  + Web Application
  + Database
* Validation Testing - Test software as whole
  + Web Application
  + Database
* High-order Testing
  + Web Application
  + Database

## Major constraints

Any business, product line or technical constraints that will impact the manner in which the software is to be tested are noted here.

* Development team does not have enough members to perform an accurate stress test related testing on the software.
* Software will not be connected to a host of the clients choosing until the project is complete and signed off on, so no accurate testing of page load times can be made.

# Test Plan

The end goal is for the software to be bug free. To ensure there are no defects we will spend a large amount of time testing the software. Below is the description of the testing procedure and strategy.

## Software (SCIís) to be tested

The software to be tested is identified by name. Exclusions are noted explicitly.

* Login
  + Login will be tested using multiple user accounts. Both Admin and Employee security levels will be tested.
* Employee User Interface
  + The employee interface will be tested.
    - User profile page will be tested to ensure the user information is displayed.
    - All data fields will be edited and we will test the submit button to make sure changes are saved to the database.
    - Users Team ranking page will be tested to see if all teams are ranked in the correct order.
    - We will also test if the user has the ability to click on their team to view their teams analytics.
    - We will test to see if team history is displayed when user selects page.
* Manager Admin Interface
  + The Managers admin interface will be tested.
    - Manage team page will be tested to see if add team button creates a new team.
    - Manage team page will be tested to see if clicking add on an employee will add them to a team.
    - Manage team page will be tested to see if clicking remove on an employee will remove them from a team.
    - Manage employee page will be tested to see if on load a list of all employees are displayed.
    - Manage employee page will be tested to see if clicking add routes to a blank employee information page to create a new employee.
    - Add employee page will be tested to see if clicking submit after entering an employee's information inserts the data in the database.
    - Edit employee page will be tested to see if clicking submit after updating an employee's information updates the data in the database.
    - Manage employee page will be tested to see if clicking remove on an employee removes the user from the database.
* RankUpdater
  + Test to see if the function creates a record for active teams weekly
  + Test if teams that are added also have ranks generated for them
  + Test to see if the ranks updated with the correct score information
    - totalScore, actualScore, totalWeightedScore, and actualWeightedScore
  + Test to see if each company rank record are updated

## Testing Strategy

## Unit Testing

For unit testing we will be testing each component individually. We will be using black box testing method. Each component will be tested by inputting data and checking the output. We will do this to find any possible bugs/defects in the software.

## Integration Testing

For integration testing we will be testing each component before committing the remote branch to the master branch. When a functionality is complete and has been tested on the remote branch, it will be reviewed by two other team members. Once both members approve the code, it will be pushed to the master branch and tested to see if the functionality integrates properly with the software.

## Validation testing

For validation testing we will work with the client to ensure the software is up to the clients requirements. We will go through every feature with them and have them test out the software the way they will use it.

Black box testing will be used for our validation testing, since the client does not have the ability to see what's happening in the backend. Data will be inputted and the output will be analyzed for accuracy.

## High-order testing

For high-order testing we will be using different test methods to test different conditions.

* Stress Testing
  + For stress testing we will be testing at varying frequencies with abnormal data inputs in large sums to validate the braking point of the software.
* Performance Testing
  + For performance testing we will be testing the range of performance to know the softwares compatibility and functionality.
* Security Testing
  + For security testing we will be testing the two-level user authentication at login as well as SQL injection testing.

## Testing resources and staffing

Information on testing resources and staffing can be found in section 3.3 of this document.

## Test work products

The work products produced as a consequence of the testing strategy are identified in section 3.4 of this document. Please refer to section 3.4 for further information on test work products.

## Test record keeping

Mechanisms for storing and evaluating test results are specified in section 3.5 of this document. Please refer to section 3.5 for further information regarding test record keeping.

## Test metrics

Our testing metrics will be dependant on the quantity of test cases and the quality.

We will use a simple percentages to determine how successful testing is.

We will divide passed test by all tests.

This will indicate the percentage of tests we have accomplished.

## Testing tools and environment

A description of the test environment, including tools, simulators, specialized hardware, test files, and other resources is presented here.

Testing environment will be using visual studio for back end. The front end testing will be using the developer tools built into modern browsers e.g chrome F12 tool. The front end testing environment will be done using chrome, edge and firefox.

We will be simulating the workload to the website and database using sample data provided by the client, the sample data is based on real data from companies that use their software.

The sample data includes 5 companies with many FixedJobs (teams), Tags (assets) and postedJobs (ranking data).

The ranking data will be simulated by either asking the client to update posted jobs daily or adding data in manually to the database. This will show the ranking system with different input.

## Test schedule

A detailed schedule for unit, integration, and validation testing as well as high order tests is described.

* Unit Testing
  + 07/11/2019 - 07/14/2019
* Integration Testing
  + 07/15/2019 - 07/17/2019
* Validation Testing
  + 07/28/2019 - 07/29/2019
* High-order Testing
  + Stress Testing
    - 07/15/2019
    - 07/28/2019
  + Performance Testing
    - 07/28/2019
  + Security Testing
    - 07/11/2019 - 07/18/2019
    - 07/28/2019

# Test Procedure

This section describes as detailed test procedure including test tactics and test cases for the software.

## Software (SCIís) to be tested

Please refer to section 2.1 of this document for information on the software to be tested and any exclusions.

## Testing procedure

The overall procedure for software testing is described.

## Unit test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Type** | **Description** | **Input** | **Expected Output** |
| 1 | Black Box Testing | Testing log in. | Username: admin  Password: password | User is logged in. |
| 2 | Black Box Testing | Testing if admin can create a team. | Admin creates team. | Team is created. Team is saved into database. Admin is now able to edit team. |
| 3 | Black Box Testing | Testing if admin can deactivate a team. | Admin deactivated team. | Team is deactivated. Team is not shown on ranking pag, and does not show up as part of active teams. |
| 4 | Black Box Testing | Testing if admin can add employee to team. Through team management page | Admin adds employee to team. | Employee is added to team. Changes made to team are saved in database. |
| 5 | Black Box Testing | Testing if admin can remove employee from team. Through team management pag. | Admin removes employee from team through team management page. | Employee is removed from team. Changes made to team are saved in database. |
| 6 | Black Box Testing | Testing if admin can create employee account. | Admin creates employee account. | Employee account is created. Credentials are generated for employee. Credentials are saved in database and sent to user email. |
| 7 | Black Box Testing | Testing if admin can deactivate employee account. | Admin deactivates employee account. | Employee account is deactivated. Credentials are revoked. |
| 8 | Black Box Testing | Testing if admin can create a team with the same name. | Admin creates team with same name. | Account is not created. |
| 9 | Black Box Testing | Testing if admin can add the same employee to a team twice. Via team management page | Admin selects employees to add. | Team members do not appear |
| 10 | Black Box Testing | Testing if admin can change metrics. | Admin Changes weight of asset. | Weight of asset is changed. Metrics are updated. |
| 11 | Black Box Testing | Testing if user can view team history. | User selects view team history. | Team history is displayed to user. User can view all their ranking history. |
| 12 | Black Box Testing | Testing if user can view weekly ranking. | User selects view weekly ranking. | Weekly ranking is displayed to user. User can see what ranking position they are in and the ranking of all the other teams. |
| 13 | Black Box Testing | Testing if user can view all time ranking. | User selects view all time ranking. | All time ranking is displayed to the user. user can view their all-time ranking and the all-time ranking of the other teams. |
| 14 | Black Box Testing | Testing if user can change their password. | User selects change password and confirms new password. | Users password is changed and saved in database. |
| 15 | Black Box Testing | Testing if user can use the same password when changing password. | User attempts to change password to existing password. | User password is not changed. |
| 16 | Black Box Testing | Testing if user can change their account information. | User makes changes to account information. | User account information is changed and saved in database. |
| 17 | Black Box Testing | Testing if user can save account information if nothing was changed. | User selects change account information and saves without making any changes. | User account information is changed and saved in database. |
| 18 | Black Box Testing | Test to see if invalid credentials are allowed. | Username:invalid  password:invalid | Error is displayed saying the credentials are invalid |
| 19 | Black Box Testing | Test to see the result of running the service function when no assets are in the table. | N/A | The service should not create a rank record unless the total score is greater than 0 |
| 20 | Black Box Testing | Test to see if deactivated teams show on the leaderboard. | Deactivate team | Deactivated teams should show on the archived leaderboard, but not new records |
| 21 | Black Box Testing | Test to see if a deleted employee can still log in | Delete employee and then attempt to log in with the credentials | The “deleted” employee should get an error that is the same as invalid credentials |
| 22 | Black Box Testing | Test to see if job reports are duplicated/recounted into ranking for the week when past 12 and the service runs | Have the service run at 12, then have a jobreport be entered while its running. It should have a timestamp past 12 | The job reports that come after 12 should not be added into the score of rank but saved for the next day. |
| 23 | Black Box Testing | Test to see if employee account can be created twice. | Create duplicate employee account. | An error is thrown. “Employee already exists”. |
| 24 | Black Box Testing | Test to see if deactivated employee shows in a team list. | Deactivate an employee and view the team list. | Employee should not be present in the team list. |
| 25 | Black Box Testing | Testing if user receives email containing username and password once their account is created. | Create new employee account. | Employee should receive an email containing their username and password. |
| 26 | Black Box Testing | Testing log out. | Click logout button | User is logged out |
| 27 | Black Box Testing | Testing if admin can add employee to team. Employee management page | Admin adds employee to teams. | Employee is added to teams. Changes made to team are saved in database. |
| 28 | Black Box Testing | Testing if admin can remove employee from team. Through employee management paeg. | Admin removes employee from team through team management page. | Employee is removed from teams. Changes made to team are saved in database. |
| 29 | Black Box Testing | Testing if admin can add the same employee to a team twice. Via employee management page | Admin selects teams to add employee to. | Teams the employee is a member of do not appear |
| 30 | Black Box testing | Test if user can import tags (assets) from api | Admin select asset to import through asset management | Assets are imported |
| 31 | Black box testing | Test if tags can be duplicated | Admin selects to import assets | Only assets that are not in database are shown |
| 32 | Black box testing | Test if users created by admin are inserted to correct shard and uses correct shard when accessing database | Create user account | User has correct companyId, and is directed to the correct shard |
| 33 | Black box testing | Test if users can access different shards | Login as user | User is only shown data from shard he is associated with |
| 34 | Black box testing | Employees should only be able to see ranks and user profile, test if he can see other menus | employee logs in and opens navigation bar and | Rank menu and user profile buttons are displayed and are functional |
| 35 | Black box testing | Admin should be able to see all the pages for admin role including: team management, user management, asset management, ranking menu | Admin logs in and opens navigation bar | Navigation bar displays  team management, user management, asset management, ranking menu buttons and are functional |

## 

## Integration testing

The integration testing procedure is specified.

## Testing procedure for integration

For integration testing we will be testing the software components individually and then testing the software to make sure it has been integrated properly.

## Stubs and drivers required

Each component will be tested with a test function (stubs) which will be used to test the component without having all the functions of the software available.

## Test cases and their purpose

Each test case will be tested once the software has been integrated to the master branch. We will do this to ensure the software still functions the same once a new functionality has been integrated. We will also test to make sure the software retrieves data from Invisi-tag once the software is up and running.

## Expected results

## Validation testing

The validation testing procedure is specified.

## Testing procedure for validation

For validation testing we will be testing the software with our client to make sure the software is up to the clients requirements. All test cases will be tested once the software is fully complete.

## Expected results

We expect all tests to have positive results. All functionalities should satisfy the clients requirements. All buttons should work and function as specified. All data should be stored correctly in the database.

## Pass/fail criterion for all validation tests

All test cases should pass to show the client the validity of the software being presented to them. Any issues presented will be fixed and tested again. Any requirements not met will be added and tested in front of the client.

## High-order testing (a.k.a. System Testing)

The high-order testing procedure is specified below. For high-order testing we will be using different test methods (Stress Testing, Performance Testing, and Security Testing) to test different conditions.

## Security testing

For security testing we will be testing the two-level user authentication at login as well as SQL injection testing.

## Stress testing

## For stress testing we will be testing at varying frequencies with abnormal data inputs in large sums to validate the braking point of the software.

## Performance testing

For performance testing we will be testing the range of performance to know the softwares compatibility and functionality.

## Pass/fail criterion for all validation tests

All high order tests should pass without any issues. Performance of the software will be dependent on the host's server response time.

## Testing resources and staffing

Since this is a web application and requires an internet connection and browser to work, we will be using our own laptops and mobile devices to test the software. All members of the development team will be testing the software.

## Test work products

The work products produced as a consequence of the testing procedure are identified.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test #** | **Test Description/Basis** | **Test Conditions** | **Test Coverage** |
| 1 | Testing log in. | Pass | Tested |
| 2 | Testing if admin can create a team. | Pass | Tested |
| 3 | Testing if admin can delete team. | Pass | Tested |
| 4 | Testing if admin can add employee to team. | Pass | Tested |
| 5 | Testing if admin can remove employee from team. | Pass | Tested |
| 6 | Testing if admin can create employee account. | Skip | Not Tested |
| 7 | Testing if admin can deactivate employee account. | Skip | Not tested |
| 8 | Testing if admin can create a team with the same name. | Skip | Not Tested |
| 9 | Testing if admin can add the same employee to a team twice. | Skip | Not Tested |
| 10 | Testing if admin can change metrics. | Skip | Not Tested |
| 11 | Testing if user can view team history. | Pass | Tested |
| 12 | Testing if user can view weekly ranking. | Pass | Tested |
| 13 | Testing if user can view all time ranking. | Pass | Tested |
| 14 | Testing if user can change their password. | Skip | Not Tested |
| 15 | Testing if user can use the same password when changing password. | Skip | Not Tested |
| 16 | Testing if user can change their account information. | Skip | Not Tested |
| 17 | Testing if user can save account information if nothing was changed. | Skip | Not Tested |
| 18 | Test to see if invalid credentials are allowed. | Pass | Tested |
| 19 | Test to see the result of running the service function when no assets are in the table. | Skip | Not Tested |
| 20 | Test to see if deactivated teams show on the leaderboard. | Pass | Tested |
| 21 | Test to see if a deleted employee can still log in | Pass | Tested |
| 22 | Test to see if job reports are duplicated/recounted into ranking for the week when past 12 and the service runs | Skip | Not Tested |
| 23 | Test to see if employee account can be created twice. | Skip | Not Tested |
| 24 | Test to see if deactivated employee shows in a team list. | Skip | Not Tested |
| 25 | Testing if user receives email containing username and password once their account is created. | Skip | Not Tested |

## Test record keeping and test log

We will be using a table for storing and evaluating test results. The test log is used to maintain a chronological record of all tests and their results. Below is an example of the table that will be used to log all test.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test #** | **Test Description** | **Test Results** | **Date Tested** |
| 1 | Testing login. | User login successful. | 07/11/2019 |
| 2 | Testing if admin can create a team. | Admin successfully created a team. | 07/11/2019 |