NASA EVA Gamification

GAME DESIGN

PHASE II

Prepared By

Laura Addiego

Samia Alam

Kelli Corey

Charles Milk

Adeola Odusola

Hung Pham

Table of Contents

Revision Table

Version Number	Description of Change	Author	Date
1.0	Initial Creation of Document	Samia Alam	6/28/18
1.1	Added database requirement F54 to Requirements Mapping section	Charles Milk	7/21/18

Introduction

Purpose

The purpose of this software design document is to describe the architecture and system design of NASA EVA Gamification project phase 2. The document describes component design, data design, user interface design, and requirement mapping of the features that will be developed during the project. The intended audiences are developers, designers, and testers.

Scope

The primary focus of this phase of NASA EVA gamification project is to build an extension that provides a consolidated infrastructure, allowing for the addition of gamification elements to the NASA Wiki environment. This phase of the project will focus on three areas of enhancements: point measurement, badge assignment, and a leaderboard. Users will be given points for contributing to the mediawiki. Phase 2 will only focus on earning points for adding a new page or editing an existing page. Different badges will be earned based on different amount of point accumulation. The overall points, contributions, badges, and current position among other users will be viewable through the leaderboard that shows the relative position of the user. Overall, the enhancements will fulfill different aspects of gamification including encouraging and motivating users to contribute more.

Document Overview

This document describes the system architecture, Software design, data design, user interface design, and requirement matrix associated with Phase 2 of NASA EVA Gamification project.

Software Architecture Overview

The phase 2 of NASA EVA gamification project architecture has four main components:

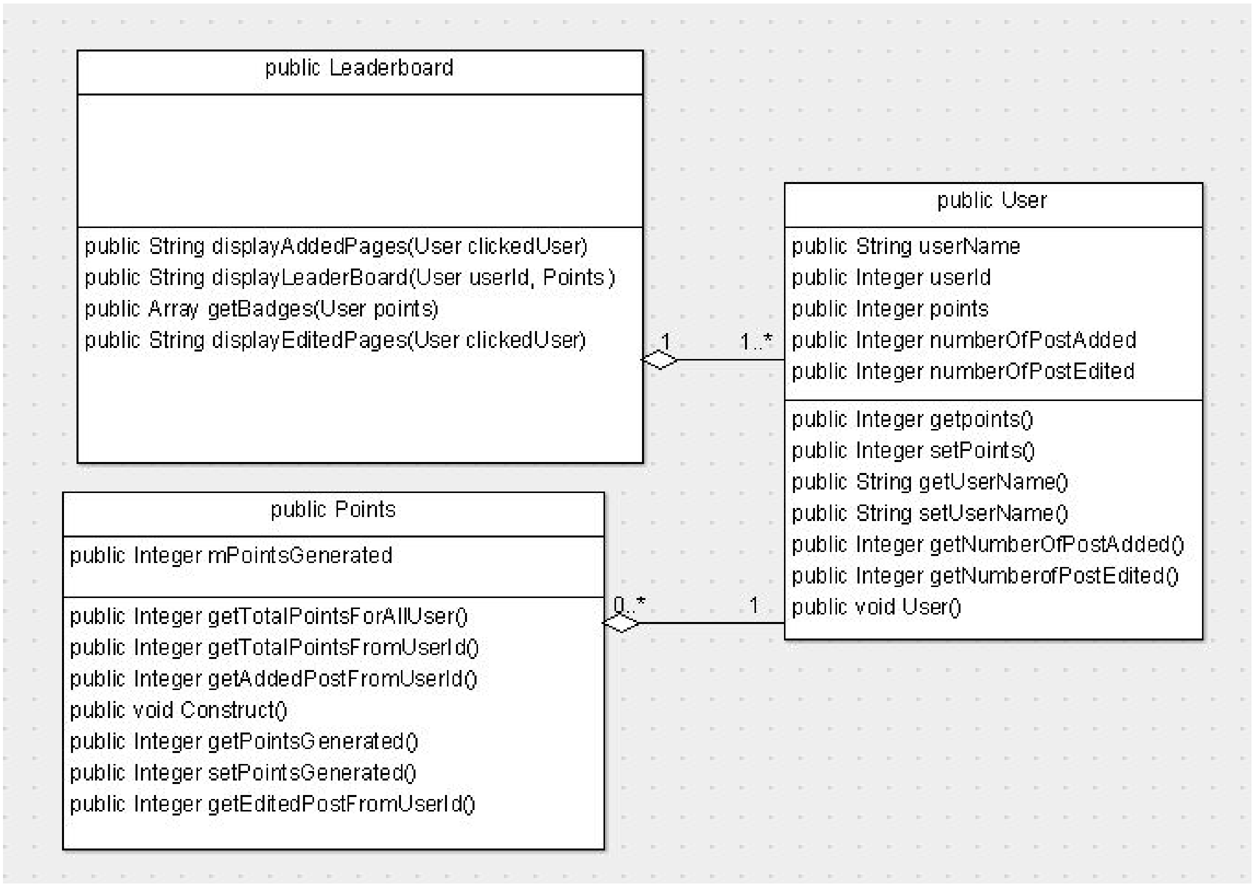
- Updating the gamification database table
- Point Assignment Configuration
- Badge Configuration based on point accumulation
- Scaled Leaderboard creation

Data Design

All MediaWiki data will be stored in specified tables within MariaDB database. See the database design documentation for further details.

Software Architecture Design

This software architecture design provides a basic concept of different classes that are interdependent on each other to fulfill the expected functionalities defined in the requirements document.



public Leaderboard
public String displayAddedPages(User clickedUser)
public String displayLeaderBoard(User userId, Points)
public Array getBadges(User points)
public String displayEditedPages(User clickedUser)
1
1..*
public User
public String userName
public Integer userId
public Integer points
public Integer numberOfPostAdded
public Integer numberOfPostEdited
public Integer getpoints()
public Integer setPoints()
public String getUserName()
public String setUserName()
public Integer getNumberOfPostAdded()
public Integer getNumberofPostEdited()
public void User()
public Points
public Integer mPointsGenerated
public Integer getTotalPointsForAllUser()
public Integer getTotalPointsFromUserId()
public Integer getAddedPostFromUserId()
public void Construct()
public Integer getPointsGenerated()
public Integer setPointsGenerated()
public Integer getEditedPostFromUserId()
0..*
1

Software Design Description

Component Name: Point Configuration

Point configuration is the base of the gamification of NASA EVA MediaWiki. User will earn points based on their contribution to mediawiki. Point assignment will vary based on the type if action that user does. Different amount of point will be assigned to a user based on whether they have added or edited a page.

Point Configuration Design Description:

Different point value will be set for a defined add and edit action taken by user. If the action is an add, then 100 points will be added to the user. If the action taken by the user is an edit, then 50 point will get added to the user. The points will be stored in user's table associated with points.

Workflows:

- When a user creates a page, 100 points will be assigned to the user

- When a user edits a page, 50 points will be assigned to the user

Point Configuration Class name: Points

Class Description: Points class will include the attributes and functions needed for point assignments to users.

Functions:

- _Construct ($page_is_new): this function will return the points generated based on the $page_is_new value
- getPointsGenerated (): this function will get the points generated
- setPointsGenerated($points_generated): this function will set the point_generated of the point
- getTotalPointsForAllUsers(): this function will get total points generated for all users
- getTotalPointsFromUserId($id): this function will get the total points generated from a given user ID number
- getAddedPostFromUserId($id): this function will get the number of added post from a given user ID number
- getEditedPostFromUserId($id): this function will get the number of edited post from a given user ID number
- Construct(): This function will declare the global array for points.

Point configuration Pseudocode:

```
Class Points {
Declare class variable $mPointsGenerated;
Function _Construct ($page_is_new) {
        Declare inherit global array $wgPoints;
If ($page_is_new is not null)
return $mPointsGenerated = setPointsGenerated ($wgPoints[$page_is_new)]);
}
Function getPointsGenerated () {
Get points_generated for current logged-in user in the table revision from the database.
}
Function setPointsGenerated( $points_generated ) {
Set points_generated of the current logged-in user to the class variable.
}
Function getTotalPointsForAllUsers(){
Get total points generated for all users in the table revision from the database.
Returns null if no user can be found.
}
Function getTotalPointsFromUserId ($id){
Get total points generated for a given user ID number in the table revision from the database.
Returns null if no such user can be found.
}
Function getAddedPostFromUserId( $id ) {
Get total added posts for a given user ID number by calculating number of rows that points_generated = 100 in the table revision from the database.
Returns null if no such user can be found.
}
Function getEditedPostFromUserId( $id ) {
Get total edited posts for a given user ID number by calculating number of rows that points_generated = 50 in the table revision from the database.
Returns null if no such user can be found.
}
}
```

Component Name: Badge Configuration

The badge feature is added as a reward system for the users based on their contribution. Badges will be configured based on total point accumulation. This will fulfill the gamification aspect of encouraging users to increase contribution to earn different badges.

Badge Configuration Design Description:

Badges will be configured in an array. The array will hold the badge's name and its corresponding point-value. Badge's image files will be stored in "images" folder under the "NASA_EVA_Gamification" extension folder. Badge's image file name will be the same as the badge's name. So, the same value for the badge's name will be used call the image of the badge.

Workflows:

- Whenever a user access their leaderboard page, a function will calculate the total number of points the user had generated.
- Badges will be generated on the fly according to the total number of points earned.
- Users will earn a platinum badge when their total number of points earned is greater than or equal to 500 points, and while their total points left (after subtracting 500 per every platinum badge) is greater than or equal to 500.
- Users will earn a gold badge if after earning 0, 1, or many platinum badges they still have 200 points or more left, and while their total points left (after subtracting the points of earned badges) is greater than or equal to 200.

- Users will earn a silver badge if after earning 0, 1, or many platinum badges and 0, 1, or many gold badges they still have 100 points or more left.
- Users will earn a bronze badge if after earning 0, 1, or many platinum badges and 0, 1, or many gold badges, and 0 or 1 silver badges they still have 50 points left.

Badge Configuration Class name: N/A

Class Description: N/A

Component Name: Leaderboard

Leaderboard will allow the users to view their position or status in the game as they can see the name of the other users within +/-100 points of them. It will also allow them to see how many pages they have added or edited.

Leaderboard Design Description:

The design of the leaderboard would be in a table format as the desired data would be pulled by running a query. A query will be run to get the count of add, edit, and points associated with a user. The result from the query will be made visible in the leaderboard in user's profile page.

Workflows:

- When users navigate to their profile page, then can see a leaderboard which show the Add count, edit count, points earned, the user's position in the leaderboard, and other users 100 points over and below.
- When users click on the add count, they can view the name of the pages created.
- When users click on the edit count, they can view the name of the pages edited.

Leaderboard Class name: LeaderBoard

Class Description: Leaderboard class will include the attributes and functions needed to display a leaderboard in the MediaWiki profile page.

Functions:

- getBadges ($totalPoints): This function will generate badges on the fly based on points.
- displayLeaderBoard($userId,$userPoints): This function will allow displaying of different information including username, added post count, edited post count, points, and badges in the leaderboard. The user ID will be used to highlight the user's row in the leaderboard, while the user's points will be used to scale the leaderboard to show other users within 100 points above or below the user.
- displayAddedPages($clickedUser): This function displays the name of pages created by a user when the added post count of that user is clicked.
- displayEditedPages($clickedUser): This function displays the name of pages edited by a user when the edited post count of that user is clicked.

Leaderboard Pseudocode:

```
Class LeaderBoard {

Function displayAddedPages($clickedUser){

        Query database for the names of pages created by the user whose add count was clicked in
leaderboard

        Get DB rows to display

        Return list of page names in html div

}

Function displayEditedPages($clickedUser){

        Query database for the names of pages edited by the user whose edit count was clicked in
leaderboard

        Get DB rows to display

        Return list of page names in html div

}


Function getBadges ($totalPoints) {

Set $wgBadges=array('plainum'=>500, 'gold'=>200,'silver'=>100,'bronze'=>50);

Set $badgesGenerated array = empty;

foreach($wgBadges as $badgeName=>$badgeValue){

     While ($totalPoints>=$badgeValue){

          Add $badgeName to $badgesGenerated array

          Set $totalPoints=$totalPoints-$badgeValue;

              }

}

Return $badgesGenerated array

}


Function displayLeaderBoard($userId, $userPoints) {

         Set $minPointsToDisplay=$userPoints-100;

         Set $maxPointsToDisplay=$userPoints+100;

         Query DB for userIds,  usernames, add counts, edit counts, points for only users having points
between min and max points to display

        Get database rows to display

        While database row is not empty{
```

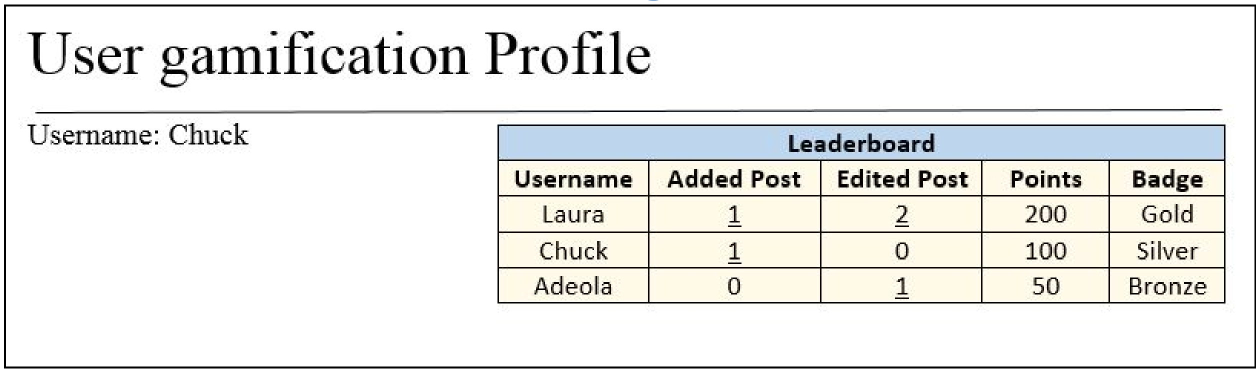
```
                        Start new html row to display
                        Highlight row if current user in  DB row equals userId in function input
                         Add username to its column
                         Add add count to its column in link format
                         Add edit count to its column in link format
                         Add points to its column
                         Call getBadges($points) using the points
                         Add badges to its column
                    }
                   Return html of leaderboard table
              }
              }
```

User Interface Design

Overview of User Interface

The User Interface will allow users to view their earned points, badges, and leaderboard associated with the EVA gamification. When user logs into their account, they will view the leaderboard in the profile page by default. The added post count will appear in link format. When user clicks on the linked value, the name of the pages the user added will be displayed. The different criterion of the leaderboard will be displayed in different columns. When user goes to a different page, the leaderboard will not be displayed anymore. The badges earned by the user will be displayed on the top right corner of the page.

Leaderboard User Interface Image



User gamification Profile

Username: Chuck

Leaderboard				
Username	Added Post	Edited Post	Points	Badge
Laura	1	2	200	Gold
Chuck	1	0	100	Silver
Adeola	0	1	50	Bronze

Requirements Mapping

The design satisfies the following business and functional requirements:

Regulatory Requirements

ID	Business Requirements
R5	All mediawiki gamification related metadata and records should be managed in MariaDB or MySQL
R4	All Code and extensions should be written in PHP

Non-Functional Requirements

ID	Non Functional Requirements
NF1	Configuring any variables should be allowed through a config PHP file

Functional Requirements

ID	Component	Functional Requirements
F43	Points	Users shall earn 100 points when a page is created
F44	Points	Users shall earn 50 points when a page is edited
F45	Badges	User with 50 total points shall earn a bronze badge
F46	Badges	User with 100 total points shall earn a silver badge
F47	Badges	User with 200 total points shall earn a gold badge
F48	Badges	User with 500 total points shall earn a platinum badge
F49	Badges	Badges shall be displayed on top of the user’s profile page in a tabular form with the earned badge images
F50	Leaderboard	Add count, edit count, points earned, and the user’s position among others will be displayed in leaderboard
F51	Leaderboard	Other users with 100 points more or less than the current user will be displayed in the leaderboard
F52	Leaderboard	Users shall be able to click on the add or edit count to view the name of the pages created or edited
F53	Leaderboard	Leaderboard will only be visible in user profile page

F54	Database	The database shall track the number of points generated each revision. This information shall be in the form of a 4 digit wide unsigned integer which cannot contain a null value.